302-EMD-001

ECS Maintenance and Development Project

Software Maintenance and Development Plan for the EMD Project

October 2003

Raytheon Company
Upper Marlboro, Maryland

Software Maintenance and Development Plan for the EMD Project

October 2003

Prepared Under Contract NAS5-03098
CDRL Item #002

RESPONSIBLE AUTHOR

Janine Smith-Carlisle /s/ 10/31/2003

Janine Smith-Carlisle, Deputy, Custom Code Maintenance Date
ECS Maintenance and Development Project

RESPONSIBLE OFFICE

Art Cohen /s/ 10/31/2003

Art Cohen, Manager, Custom Code Maintenance Date
ECS Maintenance and Development Project

Raytheon Company
Upper Marlboro, Maryland

302-EMD-001

This page intentionally left blank.

Preface

This document is a formal contract deliverable. It requires Government review and approval within 20 business days. Changes to this document will be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The EMD Project Office
Raytheon Company
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5301

Revision History

Document Number	Status/Issue	Publication Date	CCR Number
302-EMD-001	Revision -	October 2003	03-0727

This page intentionally left blank.

Abstract

The EMD SDPS Software Maintenance and Development Plan (SMDP), CDRL item 002, DID EMD-SMDP-2, defines the steps by which the development and maintenance of EMD SDPS software will be accomplished and the management approach to software development and maintenance. The SMDP addresses software processes, methods, organizational responsibilities, tools, configuration management, software quality, metrics, and other activities relevant to accomplishment of the EMD SDPS statement of work. The SMDP describes software development processes at a summary level and makes extensive reference to the collection of EMD SDPS Project Instructions (PIs). The PIs provide details for: 1) processes, such as metrics collection and inspections; and 2) project standards, such as the format and content for software development files (SDFs) and coding standards. The SMDP also provides a catalog of development, test, and delivery services that can be applied to software development and maintenance. The intent is for this document to provide the overall high-level process and the PIs and Work Instructions (WIs) to provide the detailed instructions on how EMD SDPS executes this process.

Keywords: software, process, development, maintenance, training, metrics, standards, COTS, deployment, configuration management

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1 Identification 1-1
1.2 Scope of Document 1-1
1.3 Document Overview 1-1
1.4 Relationship to Other Documents 1-3
1.5 Review Cycle 1-3

2. Related Documentation

2.1 Parent Documents 2-1
2.2 Applicable Documents 2-1

3. Software Organization and Resources

3.1 Software Organization 3-1
3.1.1 Software Organization Roles and Responsibilities 3-1
3.1.2 Inter-discipline Coordination 3-2
3.1.3 Software Training 3-3
3.2 Resources for Software Development 3-4
3.2.1 Software Engineering Environment 3-4
3.2.2 Software Development Library 3-8
3.2.3 Coding and Design Standards 3-8
3.2.4 Target Systems 3-8

4. Software Project Management

4.1 Planning Activities 4-1
4.1.1 Estimation 4-1
4.1.2 Pre-Planning 4-2
4.1.3 Detailed Planning 4-3

4.2 Decision Analysis and Resolution 4-5
4.3 Risk Management 4-5
4.4 Software Metrics 4-6
4.5 Supplier Management 4-8
4.6 Configuration Management 4-9
4.6.1 Configuration Identification 4-10
4.6.2 Configuration Control 4-10
4.6.3 CM/DM Library and Electronic Repository 4-10
4.6.4 Software Configuration Management and Release Process 4-11
4.6.5 Configuration Status Accounting 4-11
4.6.6 Configuration Audits 4-11
4.7 Software Product Evaluation 4-11
4.8 Reviews 4-12
4.8.1 Program Daily Status Reviews 4-12
4.8.2 Software Senior Management Review 4-13
4.8.3 Peer Reviews 4-13
4.8.4 Software technical reviews 4-13
4.9 Project Process Improvement 4-15
4.10 Software quality engineering 4-15

5. Catalog of Services

6. Software Development Process

6.1 Development Life-Cycle Approaches 6-1
6.1.1 Formal Development 6-2
6.1.2 Incremental Development 6-2
6.1.3 OSS Development 6-2
6.2 Systems requirements development and management 6-4
6.2.1 Software requirements management 6-5
6.3 Software Architectural/Preliminary Design 6-5
6.4 Software Detailed Design 6-6
6.5 Software Documentation 6-6
6.6 Software Code and Unit Test 6-7
6.7 Software Integration and Test 6-8
6.8 Testing Approaches 6-8

6.8.1 Regression/Fault Recovery Test 6-8
6.8.2 Performance Verification Testing 6-9
6.8.3 Acceptance Test 6-10
6.9 Deployment Options 6-11
6.9.1 Engineering Software 6-11
6.9.2 Test Executables 6-11
6.9.3 Patches 6-12
6.9.4 Drop/Release 6-12
6.9.5 Transition/Training 6-12
6.10 Process Variances 6-13

7. Corrective Action Process/Non-Conformance Reports (NCR)

7.1 Prioritization of NCRs 7-1
7.2 NCR Process 7-1

8. COTS Maintenance/Insertion

8.1 Key COTS Software Upgrade Processes 8-2
8.1.1 COTS Software Upgrade Analysis Phase 8-2
8.1.2 Readiness and Planning Phase 8-3
8.1.3 COTS Software Engineering Phase 8-3
8.1.4 COTS Software Verification Phase 8-4
8.1.5 COTS Software Review and PSR Phase 8-4
8.1.6 COTS Software Deployment Phase 8-4
8.2 COTS Test Executables (TEs) 8-4
8.3 COTS Problem Resolution 8-5
8.4 COTS Software Tools and Configuration Control Methods 8-5
8.5 COTS Software Metrics 8-5

Abbreviations and Acronyms

List of Tables

1-1. Mapping of CDRL Requirements to SMDP Document 1-2
3-1. SDPS Maintenance and Development Tools 3-5
4-1. Software Performance Metrics 4-6
5-1. System Development Services enable cost-effective deployment of new capabilities 5-1

6-1. Software Design Documentation 6-7

List of Figures

3-1. EMD/ECS ESD Task 1 Organization 3-1
4-1. The Raytheon Risk Management Methodology. 4-5
6-1. Development Tasks and Artifacts 6-1

1. Introduction

1.1 Identification

This Software Maintenance and Development Plan (SMDP), Contract Data Requirements List (CDRL) Item 002, whose requirements are specified in Data Item Description (DID) EMD-SDMP-2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) Maintenance and Development (EMD) Project, Contract NAS5-03098.

1.2 Scope of Document

The EMD Science Data Processing Segment (SDPS) SMDP outlines the steps by which the development and maintenance of EMD SDPS software will be accomplished and the management approach to software development. The SMDP addresses software processes, products, methods, organizational responsibilities, tools, configuration management, software quality, and other activities relevant to accomplishment of the EMD statement of work. Overall, the plan for EMD SDPS software development consists of several documents:

- The EMD SDPS SMDP – discusses software development processes at a summary level
- RAYTHEON policies and directives (available on the Raytheon web page) – describes practices that apply to the Raytheon business unit
- EMD Project Instructions (PIs) and Work Instructions (WIs) and Landover Facility Policies (available on the ECS Internal Server) – provide details of how Custom Code Maintenance and other processes on the EMD project are executed
- Related Project Documentation (listed in Section 2 of this document) – provide additional information about the EMD Project, the software product, and related processes
- Baselined schedules and budgets maintained for the EMD project (available from the Program Office or Program Controls Department) – provide up to date status regarding the cost and schedule of software products under development.

These documents are applicable to all software development processes and standards on the EMD SDPS project unless a formal waiver identifying any deviation or exception is documented and approved. Note that EMD PIs and WIs take precedence over RAYTHEON policies and directives since the EMD PIs reflect the tailoring of Integrated Product Development System (IPDS) and Raytheon Software Operating Instructions (SOIs) to the EMD program, as well as specific contractual obligations.

1.3 Document Overview

The purpose of this SMDP is to describe the processes that will be used for achieving program objectives, monitoring performance progress, conducting design reviews, and performing engineering management on the program.

This SMDP is organized into the following sections:

Section 1	identifies the document and specifies the purpose of this SMDP in relation to other planning documents
Section 2	lists all documents referenced in this SMDP
Section 3	describes the software development organization and resources required to execute the program
Section 4	describes the software management activities required to design, develop and test the software products
Section 5	Summarizes a catalog of available approaches for requirements, development, integration, and test
Section 6	provides detailed information about the software engineering processes, procedures and activities
Section 7	provides detailed information about corrective action processes, procedures, and activities as related to software maintenance and management of Non-Conformance Reports (NCRs)
Section 8	provides detailed information about COTS and technology insertion processes, procedures and activities
Section 9	list of acronyms definitions used in this document

Table 1-1 maps sections of this document to the specific coverage requirements called out by DID EMD-SDMP-2.

Table 1-1. Mapping of CDRL Requirements to SMDP Document

CDRL Requirement	Plan Section
Organizational Responsibilities	Sections 3.1.1, Section 3.1.2
Resources	Section 3.2
Guidelines and Processes	Section 6
Configuration Control Methods	Section 4.6
Testing/Verification	Section 6.6, Section 6.7, Section 6.8
Quality	Section 4.10
Training	Section 3.1.3
Automated Tools	Table 3-1.
Reviews	Section 4.8
Metrics	Section 4.4
Documentation	Section 6.5
Catalog of Services	Section 5

1.4 Relationship to Other Documents

This plan is related to other plans as listed in section 2. The EMD Program Manager is the process owner for this plan and is responsible for approval or rejection, both for initial and changed versions.

1.5 Review Cycle

No updates of this document are planned unless the scope of the program changes or processes described in this document change. The Program Manager will notify the project team members of changes verbally or by email.

At anytime during the execution of the program any member of the development team can recommend a change to the development process. This is done through the normal program change control procedures. The Software Task Lead reviews the recommended change taking into consideration program cost and schedule constraints. If the recommended change is implemented then this plan is updated appropriately and re-approval is required. The Custom Code Maintenance IPT Lead communicates process changes to the development team.

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which the Software Maintenance and Development Plan's scope and content are derived.

	EMD Task 101 Statement of Work for ECS SDPS Maintenance, August 2003
423-46-02	Contract Data Requirements Document for EMD Task 101 ECS SDPS Maintenance

2.2 Applicable Documents

The following documents are referenced within the Software Maintenance and Development Plan, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

104-EMD-001	Software Quality Assurance Plan
110-EMD-001	Configuration Management Plan
303-EMD-001	Hardware Maintenance and Development Plan
EMD-RMP-6	Risk Management Plan
108-EMD-001	Program Management Plan
EMD-EDP-23	ECS SDPS Documentation Package

This page intentionally left blank.

3. Software Organization and Resources

3.1 Software Organization

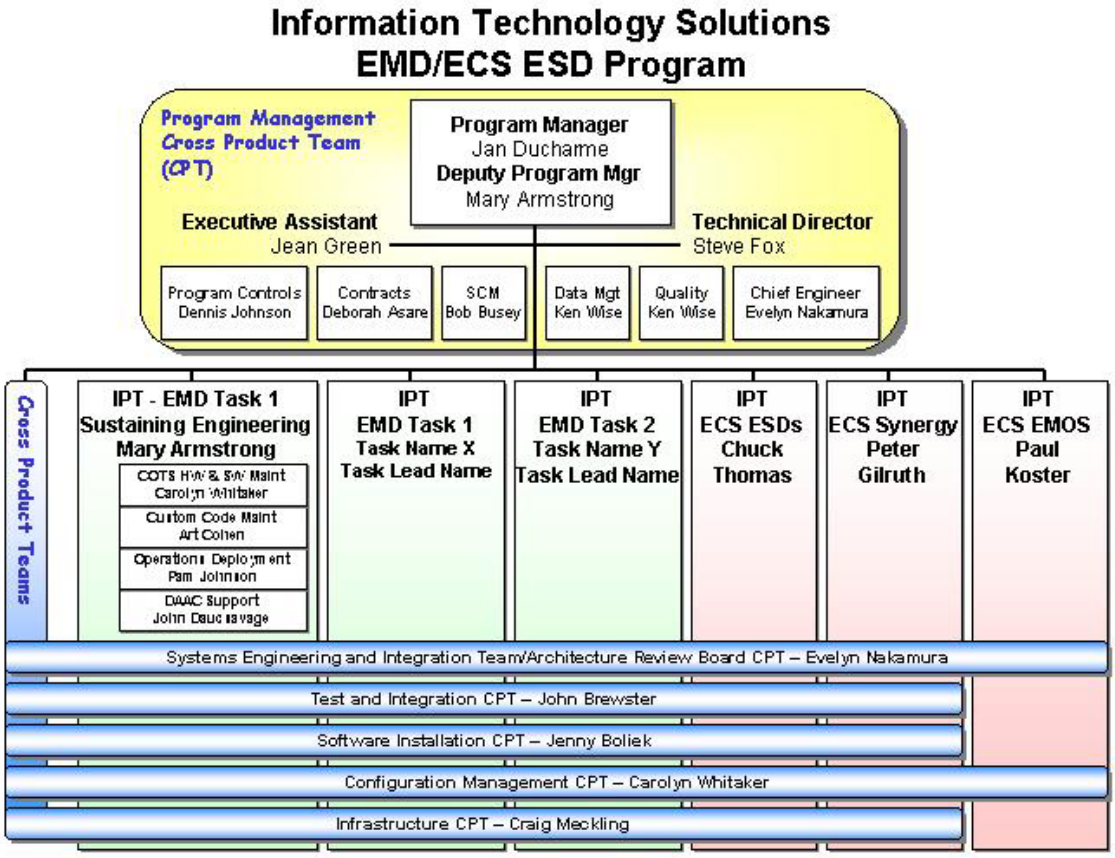

Figure 3-1. EMD/ECS ESD Task 1 Organization

The EMD Task 1 program organization depicted in Figure 3-1 supports SDPS maintenance and development and provides the framework for initiating, planning, and executing additional tasks.

3.1.1 Software Organization Roles and Responsibilities

Figure 3-1 lays out the overall formal organizational structure supporting EMD. The activities of software development span multiple organizations within the EMD SDPS project. The authority, roles, and responsibilities of the technical staff responsible for software releases within the EMD

SDPS Project are described below. The organization charts are posted on a periodic basis to the EMD Internal Server. The key organizational elements include:

- Program Management, which provides management oversight during all task life cycle phases, ensures that adequate support services are available for each task, provides the interface to the customer for cost/schedule issues, provides contractual direction to the software organization, and receives team inputs for program status reporting.
- The System Engineering and Integration Team (SEIT) and Architecture Review Board (ARB), which provide technical oversight over the SDPS architecture and design, ensure the integrity of the technical baseline across tasks, prioritize incoming work, optimize resources and schedules across tasks, perform studies and prototyping, and generate SEPs. Support Services, which provide as-needed access to key services including cost and schedule tracking, configuration management, procurement, quality assurance, computer infrastructure, contracts, and safety.
- Support Services, which provide as-needed access to key services including cost and schedule tracking, configuration management, procurement, quality assurance, computer infrastructure, contracts, and safety.
- The Sustaining Engineering Team maintains all SDPS components and provides deployment coordination across all tasks.

The Task 101 lead is a key participant in the Program Management Team (PMT) and has direct access to all of the program level resources for support activities. Through the Custom Code, COTS Hardware and Software Maintenance, Operations Deployment, and DAAC Support Teams, the Task 101 lead has direct control of all of the technical disciplines required to deliver Task 101 services.

3.1.2 Inter-discipline Coordination

Software development, like other components of EMD, is managed using both integrated product teams (IPTs) and cross product teams (CPTs).

An Integrated Product Team is an integrated multidisciplinary team of people working together to meet common objectives and organized around a product or specific service. The IPT is responsible for the charter, budget, and planning within boundaries established by the program manager. The IPT Leader is accountable for cost, schedule, product performance, and quality. As such, the IPT owns the resources to perform the work. The IPT for SDPS maintenance performs the specific service of sustaining engineering for all SDPS components. Each Task Order forms its own IPT, and may have subordinate IPTs within it to perform specialized functions. For instance, within the Sustaining Engineering IPT (Task 101), there are teams to address custom code maintenance, COTS maintenance, operations deployment, and DAAC support (see Figure 3.1).

Cross Product Teams (CPTs) are generally not responsible for developing deliverable products or a one time specific service. They normally provide similar services across many IPTs. Functions that apply to multiple tasks on the EMD Contract are managed by CPTs. Resources from the IPTs make up the CPTs as necessary to perform these functions. The following teams

will be providing support across all EMD Task Orders: Program Management, System Engineering and Integration Team/Architecture Review Board (ARB), Test and Integration, Software Installation, Configuration Management, and Infrastructure.

The organizations primarily concerned with software, the Custom Code and COTS Maintenance IPTs, work closely with the other IPTs within its task and with the CPTs in order to meet program deliverables and milestones. The integration of the various IPTs and CPTs are detailed in Section 4, Section 5 and Section 6 of this document.

3.1.3 Software Training

This section addresses the requirements of NASA-STD-2100-91, NASA-DID-M200, Development Activities Plan, Section 7, Training for Development Personnel Planning. The requirements of Section 7 include:

- Identifying the personnel requiring training,
- Identifying the types of training by categories of personnel, and
- Identifying the plan for the conduct of training.

A Raytheon training committee exists to provide training to projects. EMD SDPS maintains representatives on this committee and submits Raytheon training needs based on the types of training beneficial to EMD SDPS and the needs of the individuals on the project. For example, software developers can sometimes benefit from training on use of software tools, C++ and/or Java, object-oriented design, etc. Training needs are assessed at the department level based on the proficiency of individuals in the department in their jobs, anticipated company business directions, and individual career goals. Each department's representative provides these assessments to the training committee. The training committee then either brings in training or schedules the individuals for vendor training depending on the circumstances. The EMD SDPS representatives to the training committee are responsible for providing accurate and timely information about their training needs. They are also responsible for gathering feedback from the EMD SDPS departments that they represent.

This process was put into place for the following reasons:

- Ensure that training activities are planned,
- Provide training for developing skills and knowledge needed to perform management and technical roles,
- Ensure that individuals in the organization receive the training necessary to perform their roles.

Both process and tool oriented training requirements are identified and coordinated by the Raytheon training team with the help of the EMD SDPS representatives. As additional training curriculum is identified, the training curriculum is updated.

A training administrator is responsible for the training program. The roles and responsibilities of the training administrator include coordination with instructors, scheduling resources for each course, notifying the representatives of the training committee of course details, ensuring that materials are available and distributed on time, collecting feedback from each course (course evaluations), and maintaining records of course conduct.

3.2 Resources for Software Development

The purpose of this section is to describe the software organization and how it interacts with other project teams or IPT(s). It describes the resources necessary to perform the software development processes as defined by the plan. Resources include adequate funding and time, appropriate physical facilities, skilled people, and appropriate tools.

3.2.1 Software Engineering Environment

The ECS Development Facility (EDF) provides all of the equipment and facilities necessary for the development and maintenance of EMD software.

Each developer has a dedicated workstation in his or her workspace. This workstation provides access to all of the tools required to perform software maintenance, including editors, ClearCase, DDTS, the Software Turnover Tracking System (STTS), Purify, and SDPS system documentation. These tools as well as others used to monitor EMD are summarized in Table 3-1.

In addition to his or her individual workstation, a developer also has access to subsystem development test platforms. These platforms are used to execute components of the system not yet ready for integration—for example, components undergoing unit test. These platforms also are used to host instances of databases and COTS software products for prototype or evaluation purposes.

The Configuration Management and Infrastructure teams maintain configuration management control over all platforms used for compiling and building the software. During periods when upgrades in operating system patches or certain COTS products (such as the RogueWave class libraries) are being integrated, it will be necessary to divide the compile and build platforms into two groups, one with the old COTS software baseline and the other having the new baseline. EMD tools and procedures are used to ensure that all of the builds (developer and software CM) are performed using the appropriate platforms. The custom software source code is maintained within ClearCase, which is hosted on a suite of platforms dedicated to this task. These platforms are carefully tuned to provide maximum performance for the EMD ClearCase implementation.

The Software Integration Lab provides a key resource for EMD software maintainers and developers. The lab provides multiple software instances of the full SDPS system, referred to as modes, implemented on multiple separate strings of hardware. There are two hardware strings, each able to support three modes. In addition, two modes of the SDPS software at the System Monitoring Center (SMC) will be implemented in the lab. The lab is a shared resource, capable of running multiple software baselines. Each baseline typically will be allocated two or three modes so that multiple problems can be investigated independently and concurrently. The development baseline is hosted in two or more modes whenever long term integration or development tasks are underway.

The Software Integration Lab is used for debugging problems, testing fixes, integrating fixes at the subsystem and system level, and regression testing. The lab modes are configured to execute the currently built baseline under software CM; this baseline will be refreshed automatically from the overnight builds. With the permission of the lab lead, developers are also able to execute components of the system within a mode from builds they performed in their personal views, so that they can compare baseline performance to the behavior resulting from their

changes. The lab also provides a collaborative environment where developers from one or more subsystems can work together to solve particularly complex problems.

The EMD Infrastructure CPT provides infrastructure services for the maintenance environment, such as incremental and full system backups, database administration, system administration, and implementation of approved upgrades.

In addition to the hardware environments described above, the EDF also provides a robust array of tools. Table 3-1 lists the major tools used to support custom software, COTS software, and COTS hardware maintenance and development for SDPS. Some of these tools are used across all SDPS support activities.

Table 3-1. SDPS Maintenance and Development Tools (1 of 4)

Tool	Description
ABC++	A custom Unix tool for extracting HTML and RTF documentation from C++ programs to generate documentation and facilitate browsing.
Acrobat Distiller	COTS Product used to convert files to pdf
Acrobat Reader	COTS Product Used for reading pdf files
Apache Ant	Software build tool - JAVA
CDMTS/ECM	COTS Foxpro based Change Management tool used to process and manage CCRs. Engineering Change Manager (ECM) is a planned web-based replacement to manage CCRs and MRs.
Common Network Tools	System Activity Report, Show the Top Processes, Multi Router Traffic Grapher, Show network status, Show tape drive status
COTS Compatibility Matrix	Custom tool for tracking COTS upgrades
Crystal Reports	COTs used for Remedy reports development
DB Artisan	Database software development tool (Oracle)
DBX	DBX is a very useful COTS debugger for tracking down errors in our custom code. It is able to track the execution of the program line-by-line in the source code and report the status of every variable. Dbx is provided as a standalone binary for SGI and as part of the WorkShop package for Sun.
DDTS	Problem tracking tool
DeliveryTool	Prepares and delivers all ClearCase custom code and delivers COTS S/W
ECS Assist	Custom tool for installing SDPS custom software
Excel	Used in generating Sustaining Engineering Metrics and the OPS Priority List
Forcheck	Development support tool
Forte compilers	Compiler

Table 3-1. SDPS Maintenance and Development Tools (2 of 4)

Tool	Description
Forte for Java	Development support tool
GNU tar	A software archiving and extracting tool required for certain freeware and shareware products
HP C compiler	Compiler
HP FORTRAN compilers (77 and 90)	Compiler
IBM AIX C compiler	Compiler
IBM AIX XL FORTRAN compiler	Compiler
IDL for PC and UNIX	Science software development tools
ILM Tool	XRP-II based tool used to provide property management, license management, and maintenance work order tracking. Currently implementing ILM under Remedy.
InFocus	508 compliance tool
Java Runtime Environment(JRE)	Offers a reliable environment for deploying Java applications in the enterprise. The Java Runtime Environment provides the minimum runtime requirements for executing a Java technology-enabled application.
JAWS	508 compliance tool
Jprobe	Java memory analyzer
Linux C compiler	Compiler
Linux FORTRAN77 compiler	Compiler
LoadRunner	Automated GUI test driver
Micro-frame Project Manager	Scheduling and planning tool
Microsoft Access	Used to monitor and track the Landover Corrective Action Database; QA activity database (audits, evaluations and discrepancy reports) and the Corrective and Preventive Action Report (C/PAR) database.
Microsoft Excel	Used in generating Quality Assurance metrics for monthly reporting.
Microsoft Office Professional	General office tool set
MPM Connect	Used to transfer data from MPM to wInsight
MPM for Windows v. 2.1	Primary financial database project control & EVM
MRTG	Multi Router Traffic Grapher tool used to monitor the traffic load on network links
Oracle Developer 2000	Database software development tool (Oracle)
PERL for PC and UNIX	Scripting tool
Perl Scripts	Mac to Mac Gateway order, SCLI Orders, Capture Performance Data
PopChart	508 compliance tool

Table 3-1. SDPS Maintenance and Development Tools (3 of 4)

Tool	Description
Primavera Project Planner	Scheduling and planning tool
Purify	Memory analysis tool
PuTTY	A SSH, Telnet and Rlogin client for 32-bit Windows systems that provide a memory-resident agent not available with commercial secure shell. It is used to establish a secure connection between Remedy Admin PC and the Remedy Unix server
Rational Rose C++	Development support tool
Remedy ARS	Problem tracking tool
RogueWave	Rogue Wave software is a versatile C++ foundation class library, which is used throughout custom code. It provides single, multibyte and wide character support, time and date handling classes, multi-thread safe, generic collection classes, smalltalk-like collection classes.
Scripts	Test uses the following scripts to improve the quality of test results: Ingest - prep_ingest, Ingest EOC_trickle, eoc_spec_verify, Collect_all_ Log_ Files, Vital_stats, EMD Distribution Metrics
SGI C compiler	Compiler
SGI C++ compiler	Compiler
SGI Fortran compilers (77 and 90)	Compiler
SGI ProDev Workshop	Development support tool
Snapshot v3.5.1	Unix application used to capture a pictures of GUIs with a menu pulled down on a Unix workstation for documentation purposes
Software Estimation Tool	Tool used by custom code maintenance to estimate effort using SLOC as input.
Software Turnover Tracking System (STTS)	Custom tool for tracking merge requests and builds
Sybase Central	Sybase monitoring tool
Sybase PowerDesigner	Data modeling and design tool
TCL/Tk	Scripting tool
Top	Performance measurement tool
Tru64 UNIX C compiler	Compiler
Tru64 UNIX Developer's Toolkit	Development support tool
Tru64 UNIX Fortran compiler (77 and 90)	Compiler
VCATS	Custom tool for generating purchase orders
Visual Studio Professional	Development support tool

Table 3-1. SDPS Maintenance and Development Tools (4 of 4)

Tool	Description
Whazzup	A custom, system monitoring tool used to track the status of EMD modes and their custom code servers
wInsight Administrator v. 5.0	Administrator tool for wInsight
wInsight v. 5.0	IFR, PMR, VARs, Earned Value Analysis
WinZip	A PC based tool used for compressing and decompressing files
XML Spy	XML schema development tool
XRP-II and Accell	Inventory,logistics , and management tool
XV v3.0	Unix application used to capture pictures of GUIs on a Unix workstation for documentation purposes.
XVT DSC for AIX, HP-UX, IRIX, and Solaris	GUI development tool

3.2.2 Software Development Library

Working technical documentation is retained in the EMD Software Development Library (SDL) that is located at "ECS SW Development" (an Internal Server). The SDL is the repository for softcopy documentation and artifacts related to this effort. The artifacts are monitored and maintained by the Custom Code Maintenance Lead or designee, and becomes part of the baselined, retained documentation. In addition, per the Peer Reviews, hard copies are required for components of these artifacts. These hard copy components are stored in a secure area until they are moved to an offsite storage facility.

3.2.3 Coding and Design Standards

To enable quick response to Non-conformance reports and the integration of new capabilities, it is critical that developers implement the same coding and design standards. Coding standards for all custom code (C++ Java, Perl, SQL, C and Fortran) are applied as specified in the Project Instructions which are located at http://dmserver.gsfc.nasa.gov/proj_instr/sdpi_index.html, an internal server accessible to EMD project staff and NASA customers.

3.2.4 Target Systems

The target computer systems for planning purposes are Sun Unix servers (e.g., Sun Blade 150, Sun Blade 2000, Sun V880, or Sun 450) running Solaris 8, SGI servers (e.g., Origin 300, Origin 2000, or Origin 3000) running Irix 6.5.17, and PC workstations running Windows 2000 or Red Hat Linux 7.3. In addition, Science User ToolKit runs on AIX. The operating systems are updated based on vendor end-of-life requirements or specific project needs.

4. Software Project Management

This section defines the following processes to monitor and control the software project: planning activities and approval processes; metrics; software estimation, scheduling and tracking; cost and pricing review and approval; decision analysis and resolution, risk management and supplier management, configuration management, software product evaluation, and reviews (program and peer).

4.1 Planning Activities

System modifications require a Configuration Change Request (CCR), a System Enhancement Proposal (SEP) or a Task Plan Request (TPR). They typically represent studies, new custom software capabilities, hardware purchases, and documentation updates. The process works much the same as for emergency changes except for the timeline to completion.

The objectives of the planning process are to establish the scope, technical approach, resource allocations and budget for new work and to define the detailed schedule of events and dependencies. Planning is initiated when the Program Manager appoints a Planning Group or Task Lead to estimate or plan new work.

The three distinct aspects of planning are: 1. estimation 2. pre-planning and 3.detail planning. The Planning Process Project Instruction captures details of each of these planning efforts. This entire planning effort concludes between 10 and 20 working days of the request; the NASA requirement for response to a TPR is 25 working days. The work is ready for immediate execution when authorization is received from the customer.

4.1.1 Estimation

The Estimation Process has three distinct aspects:

- Developing and understanding requirements
- Developing Basis of Estimates (BOEs)
- Generating prices.

4.1.1.1 Developing and Understanding Requirements

Requirements are developed by the SEIT. The SEIT:

- Ensures that requirements are understood and allocated to appropriate software products
- Documents the software requirements to be used for estimation
- Uses the software requirements to identify and refine the list of software products to be estimated.

4.1.1.2 Developing BOEs

The Task Lead is responsible for coordinating the development of the work estimate across all WBS elements. For the software-related portion of the work, the Task Lead may appoint a

Software Estimation Lead. The Software Estimation Lead is responsible for generating the BOEs. The Software Estimation Lead (SEL) derives the BOEs using the following steps.

- Using requirements developed by SEIT, the SEL works with impacted subsystems to derive estimates pursuant to Raytheon's Estimating Software Size Landover Facility Policy.
- Software component estimators prepare Delivered Lines of Code (DLOC) estimates using the EMD Software Estimation Tool, including inputs regarding the Source Lines of Code (SLOC) to be developed, modified, or reused, as well as difficulty factors assumed. Along with the DLOC worksheet, estimators include the Basis of Estimate (BOE), a list of assumptions and constraints that are relevant to the estimate, as well as a list of risks that may affect the estimate (e.g., risk of using a particular technical approach, or of not having adequately trained staff). These estimates and files are sent to the Software Estimation Lead, who collects them into a single package for review.
- The Software Estimation Lead schedules a Work Estimate Peer Review to examine all of the estimates and difficulty factors.
- The Software Estimation Lead records the SLOC values agreed on in the Work Estimate Peer Review on the EMD Software Estimation Tool to provide the estimate of effort (hours) and schedule (days) to be used in the planning process. The estimate and the other outputs (assumptions, risks), are sent to the System Estimation lead, who is coordinating the program-wide estimate for this functionality. The EMD Software Estimation Tool is available on the ECS Project Forms web page.

The Software Estimation Lead files the SLOC/effort estimate and associated files in the software repository.

4.1.1.3 Generating Prices

The Task Lead coordinates with the program control analyst (PCA) to generate pricing. Using a detailed listing of resources by grade level, the PCA generates a pricing run. The pricing and BOE are reviewed with program management.

If revisions are required to a proposal estimate, then the Task Lead and program management ensure that adequate funding and resources are available to perform the software engineering and support for each software development effort and for software maintenance. Revisions to the software effort estimate are reviewed with senior software management. A history of software sizing estimates, effort estimates, schedules and all assumptions for the each program are retained in the Software Development Library (SDL) .

4.1.2 Pre-Planning

There are two steps in the initial planning process. The first step is to analyze and develop planning and processing inputs. The purpose of this step is to determine the best methodology for planning and/or estimating new work based on a CCR, SEP or a Task Plan Request. The second step is to review and approve planning and process inputs. The purpose of this step is to review and approve the planning and process inputs from the initial estimation effort to ensure that it is appropriate to proceed. The ARB is the principal approving authority for all pre-

planning activities and documents prior to proceeding to detail planning. If the inputs are approved by the ARB, detail planning can begin. The PM also provides authority to the ARB to approve start of low risk, start up activities.

4.1.3 Detailed Planning

Detailed planning is usually reserved for large, complex changes**.** The Task Lead organizes a team from the technical and business disciplines to establish the scope of the work, the technical approach, the required resources, and budget requirements. The Detailed Planning Process is documented in the Planning Process PI. Three process models are used for detail planning: one time, incremental and expedited.

4.1.3.1 One Time Planning Process

The One Time Planning Process is used for a new effort where the requirements are clear; an assessment of low risk that the analysis and design phase will affect the remaining work; and also low risk that the budget and schedule constraints will influence the scope of work. A single detailed plan is developed at the start of the subtask to include all activities to be baselined. The One Time Planning Process can be tailored. Critical steps in this planning model include.

- Hold Kickoff Meeting - The purpose of this step is to organize the planning team. Assignments are described and the inputs are reviewed.
- Create Plan for the Plan - The purpose of this step is to develop a plan of the activities required to plan the task or subtask. The plan should be used to coordinate and track the status of the planning tasks as they are executed. If this process is being used to plan work that has not been approved, the plan for the plan contains the plan of activities required to complete the Basis of Estimate.
- Develop Technical Approach - The purpose of this step is to develop a technical approach for the tasks/subtasks. If this process is being used to plan work that has not been approved, then a technical approach for the Basis of Estimate is developed.
- Develop/Reassess Estimates - The purpose of this step is to develop and/or reassess an estimate. The estimate must be justifiable, consistent, and repeatable to support planning.
- Develop Resource Loaded Schedule - The purpose of this step is to develop a schedule that is loaded with resources. If the planning effort is for work that has not been approved, then a high level schedule with potential resources to perform the work is all that is needed for this step.
- Develop Budget and Resources - The purpose of this step is to allocate budget and resources for the task/subtask. If the planning effort is for work that has not been approved, then a PC Pricing file is used.
- Obtain Commitment to Plan - The purpose of this step is to obtain Management “Buy-In” to plan.

4.1.3.2 Incremental Planning Process

The Incremental Planning Process is used for new effort where the requirements are unclear; an assessment of high risk that the analysis and design phase will impact the remaining work; or high risk that the budget and schedule constraints will influence the scope of work. An incremental phased approach to planning will allow multiple phases to be planned over time. At the start of a subtask, the first phase is planned in detail and the remaining phases are planned at a high-level. Toward the end of each phase, a detailed plan is created for the subsequent phase and the high-level plan for the remaining phases is updated. This process can be tailored.

The following steps are performed when executing the Incremental Planning Process:

- Hold Kickoff Meeting - The purpose of this step is to organize the planning team.
- Create Plan for the Plan - The purpose of this step is to develop a plan of the activities required to plan the task or subtask. The plan should be used to coordinate and track the status of the planning tasks as they are executed.
- Develop Technical Approach - The purpose of this step is to develop a technical approach for the tasks/subtasks.
- Reassess Estimates - The purpose of this step is to reassess the estimate developed in the estimation phase. The estimate must be justifiable, consistent, and repeatable to support planning. The estimates are developed for the current phase that is being planned in detail.
- Develop Resource Loaded Schedule - The purpose of this step is to develop a schedule that is loaded with resources. The schedule is being developed for the phase that is being planned in detail.
- Develop Budget and Resources - The purpose of this step is to allocate budget and resources for the task/subtask. The budget and resources are being developed for the phase being planned in detail.
- Develop and Refine the High Level Plan - The purpose of this step is to develop the high level plan for the subsequent phases of the task/subtask. This step is executed for the subsequent phases of a task/subtask that have not been planned in detail. The budget for the phases in the high level plan is held in planning packages. The schedule for the phases in the high level plan is not baselined.
- Obtain Commitment to Plan - The purpose of this step is to obtain Management "Buy-In" to plan.
- Execute Phase - The purpose of this step is to execute the work planned in detailed. If there are more phases in the high-level plan, plan the next phase in detail.

4.1.3.2 Expedited Planning Process

The Expedited Planning Process is used for low risk activities that have minimal dependencies with other activities. Minimal milestones are planned. This process can be tailored. The following steps are performed when executing the Expedited Planning Process:

- Obtain Commitment to Plan - The purpose of this step is to obtain Management "Buy-In" to plan.

- Develop Budget and Resources - The purpose of this step is to allocate budget and resources for the task/subtask.

4.2 Decision Analysis and Resolution

The System Engineering and Integration Team (SEIT) is the key EMD organizational element that provides technical oversight over the SDPS architecture and design. The SEIT provides core members to the Architecture Review Board (ARB), which reviews planning inputs and technical approaches that require formal decision analysis. The ARB ensures the integrity of the technical baseline across subsystems, environments, DAACs, and EMD tasks. The CE, who is responsible for convening meetings, composition of the review board, keeping and distributing minutes, and maintaining the ARB repository of proceedings and directives, chairs the ARB.

4.3 Risk Management

A well-structured continuous risk management approach is in place that meets the guidelines of NPG 7120.5A (see Figure 4-1). Risk factors are an integral part of our planning process for system enhancements. Factors such as technical complexity, staff experience and availability, external dependencies, and COTS integration aspects are considered in costing and scheduling from the very start. As a result, potential risks are identified and addressed early in the process and tracked throughout the development process until they can be closed.

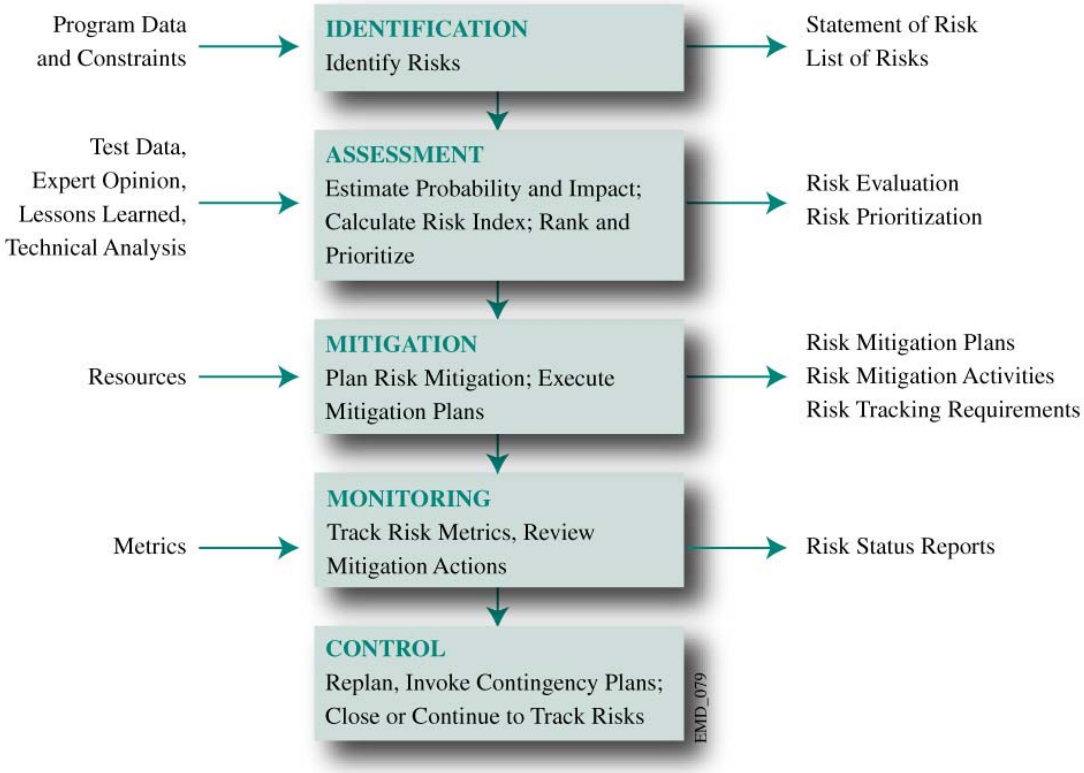

Figure 4-1. The Raytheon Risk Management Methodology

Raytheon fully documents its risk management program and methodology in a Risk Management Plan (RMP) that will be delivered within 4 months of contract award in accordance with DID EMD-RMP-6. Our methodology encompasses five major stages as depicted in Figure 4-1. It addresses technical, cost, or schedule risks, as well as those associated with methods, techniques, procedures, processes, equipment, and subcontracts related to the EMD contract. The RMP covers risk strategies involving foreign sources, unauthorized technology transfer, and includes a section on disaster recovery. The EMD RMP incorporates the Risk Management Methodology Project Instruction and Risk Assessment and Mitigation Procedures Work Instruction currently in use on EMD.

Our process allows for any individual on EMD to identify a risk. Responsibility for risk management and mitigation, on the other hand, rests with the SEIT (see Figure 4.3). The SEIT risk coordinator (RC) collects and monitors the risk inputs, and conducts risk management meetings that are attended by the PMT and chaired by the PM. New risks are assessed as to probability and impact; based on this assessment, a risk index is calculated for each risk. Risks are characterized as high, medium, or low and are ranked based on their risk index. Rankings are used to allocate resources for mitigation efforts. The RC will maintain a central repository of risk data and make it accessible to the EMD Program for planning and tracking. For each risk, a responsible individual (RI) is designated by the SEIT to lead activities related to that risk. The RI presents the status of open risks and ongoing mitigation activities weekly at customer status reviews. Risks with a high-risk index are reviewed with NASA at the monthly Program Management Review (PMR) meeting.

4.4 Software Metrics

The process of software maintenance/development can be effectively managed (monitored and improved upon) only if there is an objective means of measuring the quality of the EMD work. In order to ensure that EMD work is aligned with NASA goals and priorities, a comprehensive set of metrics has been selected. These metrics enable the EMD contractor and NASA management to evaluate and improve the quality, productivity, and effectiveness of products and services, and to measure Raytheon's performance on the EMD contract. Table 4-1 presents Raytheon's understanding of key NASA goals and summarizes our proposed metrics related to each goal. The metrics were selected to provide a quantitative measure of success. They capture the core characteristics of cost performance, schedule performance, mission success, and the quality and timeliness of deliveries to the field. Those metrics that we propose to jointly share with other stakeholders (e.g., DAACs and instrument teams) are indicated by the word "Shared" in that column.

Table 4-1. Software Performance Metrics (1 of 3)

User Satisfaction		
Metric	**Description**	**Sys Perf**
Order Response Time	Average order fulfillment response time shows the average time required to complete an order.	Shared
Severity 1 NCRs fixed by Engineering Software	Average number of days from MR to delivery for Severity 1 NCRs for engineering software (ES)	X

Table 4-1. Software Performance Metrics (2 of 3)

User Satisfaction (cont.)		
Metric	**Description**	**Sys Perf**
Severity 1 NCRs fixed by Test Executables	Average number of days from MR to delivery for Severity 1 NCRs for test executables (TEs)	X
Top 25 NCRs Fixed	Average number of days from MR to delivery for top 25 NCRs on Program Priority List	X
Sustaining Engineering		
Metric	**Description**	**Sys Perf**
Sustaining Engineering	Percentage of mission milestones achieved during the month	X
Information Flow		
Metric	**Description**	**Sys Perf**
Information Flow Down to DAACs	Percentage of patches and TEs for which the DAACs require additional information	X
DAAC Information	Rating of accuracy and consistency of basic MR information received from DAACs on a scale of 1 to 5	Shared
DAAC Top 25 Ranking	Changes in DAAC's relative ranking of top 25 NCRs	Shared
NCR Problem Resolution Effectiveness		
Metric	**Description**	**Sys Perf**
NCR Failure Rate at DAACs	Percentage of NCR fixes that failed at the DAACs	X
NCR Failure Rate During Test	Percentage of NCR fixes that failed during test	X
Installation NCRs for TEs	Number of installation NCRs generated for TEs from DAACs	X
Installation NCRs for Patches	Number of installation NCRs generated for patches from DAACs	X
Installation NCRs for COTS	Number of installation NCRs generated for COTS upgrades from DAACs	X
Number of New NCRs from Patches	Number of new NCRs resulting from patches from DAACs (break/fix ratio)	X
DAAC Effectiveness		
Metric	**Description**	**Sys Perf**
DAAC Test Executable Installation	Average time in days for DAACs to test and install a TE into operations	X-Shared
DAAC Patch Installation	Average time in days for DAACs to test and install a patch into operations	X-Shared
DAAC NCR Test/Verify	Average time in days for DAAC to test and verify NCR after receipt of patch or TE	X-Shared
DAACs Operation		
Metric	**Description**	**Sys Perf**
DAAC Ingest Performance	Number of data granules and volume ingested at the DAACs versus expected for the period	X-Shared

Table 4-1. Software Performance Metrics (3 of 3)

DAACs Operation (cont.)		
Metric	**Description**	**Sys Perf**
DAAC Distribution Performance	Number of data granules and volume distributed at the DAACs versus expected for the period	X-Shared
DAAC Production Performance	Number of data granules and volume produced at the DAACs versus expected for the period	X-Shared
System Reliability	Availability of system functions versus expected for the period (software reliability)	X-Shared
Costs/Schedule Effectiveness		
Metric	**Description**	**Sys Perf**
Schedule Performance Index	Shows the budget of work performed (BCWP) / budget of work scheduled. This provides an indicator of the efficiency of the progress being made towards the scheduled work.	X
Cost Performance Index	Shows the budget of work performed (BCWP) / actual cost of work performed (ACWP). This provides an indicator of the efficiency of the progress being made towards the estimated costs.	X

In addition, Raytheon evaluates metrics relating to software size, software staffing, computer resource utilization, fault density, software volatility, software reliability, design complexity and fault type distribution.

4.5 Supplier Management

Raytheon's Supply Chain Management (SCM) Department is responsible for subcontract and procurement management oversight. The SCM leader and subcontract administrators (SCAs) are responsible for the overall administrative management of the subcontractors and vendors supporting the EMD Program. The SCAs are responsible for the day-to-day administration of the Subcontract Agreement and task orders. The SCM leader and the SCAs are the only individuals that can authorize contractual changes to subcontract agreements. In support of EMD, the SCM staff will issue IDIQ subcontracts, similar to the prime contract, with the ability to issue CPAF, Time and Material, and Firm Fixed Price task orders. The type of task order to be issued to a subcontractor will depend on the scope and complexity of work to be performed for a specific task.

The SCAs issue the initial subcontract agreements, subsequent task orders against those agreements, and modifications to both the subcontract agreements and task orders. All subcontract agreements and task orders include all required prime contract flow-downs, comply with public law and federal regulations, and are issued in accordance with Raytheon's Government approved policies and procedures. The SCAs also oversee invoice processing, funding requirements, perform cost and price analysis on proposals, obtain technical evaluations, ensure Government consent requirements are met, perform "make or buy" analyses, conduct negotiations, and interact with the task leaders, as required.

Those subcontractors that qualify under the prime contract requirements will submit Monthly Progress Reports (EMD-MPR-10), Contractor Cost Reports (EMD-533-11), and monthly Contractor Manpower Reports (EMD-MCMR-12) to their assigned Raytheon SCA for incorporation into the prime contract deliverables, as appropriate.

The SCAs hold monthly Subcontractor Progress Review (SPR) meetings with each of the subcontractors. In addition to the SCA and the subcontractor's program manager and staff, these meetings are attended by the Raytheon EMD PM, task leaders, SCM lead, and financial lead. In these meetings, we identify issues that may affect the subcontractor's ability to perform, meet schedule, or meet costs; identify long and short term risk areas; and provide feedback or technical direction. The SCA and subcontractor's program manager meet at least once a week to ensure prompt resolution of actions items from these meetings.

The SCM leader oversees the entire subcontractor award fee evaluation process. The SCM leader ensures that each subcontractor is issued a Performance Evaluation Plan as part of the basic subcontract agreement. The SCAs, along with the task leaders, draft and issue evaluation criteria, specific for each subcontractor, prior to the start of an award fee evaluation period. At the conclusion of the award fee period, the SCA requests that the subcontractor submit its self-evaluation. This evaluation, with the criteria, is routed to the Program Management Team and task leaders for inputs. Upon receiving the comments, a Performance Evaluation Board (PEB), which includes the SCM leader, the EMD PM, task leaders, SCAs, program controls lead, and quality lead meets. The EMD Program Manager is the Fee Determining Official.

The Raytheon task leaders perform as the SCM technical representative. Within the scope given to them by the program manager, the Raytheon task leaders provide daily direction and coordination with the subcontractors that provide support to them. The task leaders ensure that activities between and among Raytheon and subcontractors' staff are coordinated and efficient.

4.6 Configuration Management

This section covers plans and processes for configuration management (CM) of EMD SDPS software. Software configuration management is the responsibility of the Configuration Management Cross Product Team, as identified in the organization structure of the ECS/EMD Program. The *Configuration Management Plan* for the EMD Program, CDRL #019, EMD document number 110-EMD-001, contains a complete description of the CM process and the services provided that support software CM requirements. These services include:

- Identification of all SDPS configuration-controlled items, including current version/release information for software and documentation.
- Configuration control and change management, including receipt, processing, review, disposition, implementation, and verification of baseline changes, including internal and external interface changes, establishment of a CCB, and management of changes flowing between the EMD and NASA CCBs. These changes may include Modification Requests (MRs) introduced as a function of EMD, as well as the standard Configuration Change Requests (CCRs).
- Management of a central CM/DM library and electronic repository, including physical and electronic retention and control of baselines for SDPS software, system hardware and software configurations, procedures, standards, and documentation.

- Implementation of an engineering release process for formal approval and CM release of all delivered SDPS software, hardware, and documentation.
- Status accounting and reporting of SDPS hardware and software information.
- Configuration audit and verification, including ensuring the integrity of hardware and software, as specified in controlled configuration documentation.

The following subsections provide a discussion of the key CM activities that are required to satisfy software CM requirements.

4.6.1 Configuration Identification

The configuration items controlled under CM are documented in EMD System Baseline Specification (905-TDA-001). This document defines the configuration items for the EMD contract, including all technical documentation, commercial off the shelf software, custom software, COTS hardware, operating systems (O/S), and O/S patches, databases, and technical documentation. Refer to Project Instruction CM-1-042, *Configuration Identification*, for configuration item definitions.

The EMD custom code components are not listed in the EMD System Baseline Specification. The ClearCase tool contains all the information about all the custom code files and versions. The Configuration Management Plan and CM project instructions describe the use of ClearCase to control source code, generate the build products, prepare, and deliver custom code software.

4.6.2 Configuration Control

Configuration control is maintained through Configuration Control Requests (CCRs) and Configuration Control Boards (CCBs). Changes are adequately defined, assessed for technical, cost, and schedule impacts by the EMD office(s) and formally considered by the appropriate CCB. Only approved changes are incorporated in the appropriate baseline.

4.6.3 CM/DM Library and Electronic Repository

CM is responsible for establishing and controlling the COTS Software Library . This library is the repository for vendor-supplied COTS software, and for COTS software that has been tailored for use on the EMD program. CM ensures that CCB-authorized material is archived and stored in the library, and that no unauthorized changes are made to established software baselines. The Data Management Office (DMO) controls hardcopy material after approval by a CCB.

CM is responsible for maintaining accountability for materials in the SDL, and for making and releasing copies to internal and external users.

The Configuration Management Plan for the EMD Project describes the process and tools for maintaining the SDL. The ClearCase tool provides for the management of the code. The CM organization maintains scripts in addition to ClearCase to provide automated controls for baseline maintenance and to help the developers.

Developers use ClearCase during the development of the software from design through unit test. After the unit test phase, a merge request is submitted and approved. Upon merging the code to the appropriate baseline, the CM group builds the software at the system level for use in the integration lab. When the release or patch is ready for delivery outside of the EDF, the CM

group delivers the release or patch using automated processes. Code or executables are only delivered to the sites through the CM organization.

4.6.4 Software Configuration Management and Release Process

EMD SDPS software follows the processes and flow described in CDRL #019, 110-EMD-001, *Configuration Management Plan*, as it migrates from the individual programmer levels, to the segment level, and then to the EMD SDPS system-level. In addition, software migrates from each EMD SDPS release to its following release in a controlled manner. The Release Notes Document, which documents the contents of a release, is an integral part of all release deliveries. Functional Configuration Audits (FCA) and Physical Configuration Audits (PCA) verify all formal deliveries. Detailed information on FCAs and PCAs can be found in the *Configuration Management Plan*, 110-EMD-001.

4.6.5 Configuration Status Accounting

Configuration status accounting consists of recording and reporting information about the configuration status of the EMD SDPS Project's documentation, hardware, and software products, throughout the Project life cycle. Periodic and ad hoc reports keep ECS SDPS Project management and NASA informed of configuration status as the Project evolves. Reports to support reviews and audits are extracted as needed. CM maintains CM Web pages. Configuration Status Accounting is described in CDRL 110-EMD-001 *Configuration Management Plan*. Referenced project instructions provide additional details on configuration status accounting.

4.6.6 Configuration Audits

Configuration auditing is the means by which management ensures that both the technical and administrative integrity of the product are being met. Real-time audits are conducted to verify that CCRs are correctly executed. These audits compare the baseline Technical Documentation to the “as built” configurations of managed EMD hosts. Differences are noted and resolved.

The audit process consists of CM self-audits and ECS SDPS Project internal audits. Formal audits are prerequisites to formal approval of the "as-shipped" configuration. They provide verification that each CI in the baseline being shipped is logically related to the corresponding CI in preceding baselines. Configuration audits (including FCAs and PCAs) are described in the *Configuration Management Plan* for EMD, 110-EMD-001.

4.7 Software Product Evaluation

The Quality Assurance organization is responsible for ensuring that EMD software work products are evaluated at various stages throughout the development lifecycle. The purpose of work product evaluation is to objectively evaluate adherence to project processes against its process description, standards, and procedures, and address non-compliance. This is accomplished via engineering and development peer review, in accordance with a documented process. For the EMD Program, software work products include requirements, interfaces, and operations concepts, documented as Tickets; design artifacts, code, unit and integration tests, and system level verification and validation tests. Quality Assurance Engineers (QAE) and other review participants are notified by the engineering organization and provided with appropriate

review materials. The QAE performs a dual role in the peer review process. First, as a reviewer, providing input on product content and quality. Secondly, as an auditor, evaluating the conduct of the peer review and its related activities with regard to adherence to applicable standards and documented procedures.

QA may monitor various test activities, including unit test and integration test demonstrations prior to turnover for system level test, as appropriate. QA also attends and monitors formal tests that may be witnessed by independent verification and validation (IV&V) representatives. QA performs audits of the test-related processes and evaluations of test artifacts, including test plans and procedures, test results and status maintained in test folders.

Quality Assurance engineers document their audit and product evaluation results in the Quality Assurance Tracking Database, which is access-restricted to QAEs. In addition, a physical records repository is maintained and includes the complete audit records, i.e., formal audit or product evaluation report, Deficiency Reports (for non-conformances), QA checklists, and other artifacts acquired as objective evidence.

Please refer to the Software Quality Assurance Plan, (104-EMD-001) for further information about the Quality Assurance organization and its activities.

4.8 Reviews

This paragraph describes program reviews and meetings for the project. Minutes of all meetings are taken, including attendance, and distributed to the review participants and other affected groups. All action items taken during reviews are tracked to closure. The status information is saved in the SDL . The minutes from reviews and a copy of the material presented are retained at least through the duration of the software development effort.

Status reviews include the following information as appropriate:

- Software schedules
- Cost
- Accomplishments, plans, issues
- Results of any audits or reviews
- Risks
- Metrics
- Action item status
- Noncompliance issues

4.8.1 Program Daily Status Reviews

Chaired by the Program Manager, this review includes the program's functional managers and support group managers: risks, schedule status, action items, and other identified items are the nominal topics. Each EMD task is reviewed weekly on a designated day. NASA representatives are invited to attend so that they can provide comment and direction on specific activities as needed.

4.8.2 Software Senior Management Review

The software engineering manager will conduct periodic Software Management Reviews; frequency will depend on the existing work level but nominally on a quarterly basis. The software manager will define the attendees and schedule the meeting, and will be responsible for the agenda and minutes.

4.8.3 Peer Reviews

The Custom Code Maintenance Lead or designee coordinates the peer review of selected artifacts during the development of the software product. The Peer Review PI (located on an internal server) is used to critique walkthroughs. The findings and decisions from peer reviews are recorded in the SDL and action items are written and tracked to closure as necessary.

Procedures for planning, conducting, and the subsequent analysis of the peer review data can be found in Peer Reviews WIs (located on an internal server). The type of Peer Review work instructions and artifacts are dependent upon the type of peer review. Detailed peer review information is included in Section 6 (Software Architectural/Preliminary Design, Software Detailed Design and Software Code and Unit Test) of this document.

4.8.4 Software technical reviews

Technical reviews include regular technical dialog with the DAACs, the instrument teams, the user community, and NASA domain experts throughout the life cycle. In addition, formal technical reviews will be required at critical phases in the development and maintenance life cycle in order to assess the readiness for proceeding to the next phase. Reviews will generate feedback that will improve the quality of future EMD products and services.

The technical reviews cover the full range of maintenance and development activities defined as follows:

- Corrective Maintenance – Changes necessitated by actual errors (i.e. 'bugs'), or design deficiencies. Corrective maintenance consists of activities normally considered to be error correction required to keep the system operational. By its nature, corrective maintenance is usually a reactive process. Corrective maintenance is related to the system not performing as originally intended. The three main causes of corrective maintenance are (1) design errors, (2) logic errors, and (3) coding errors.
- Adaptive Maintenance – Changes initiated as a result of changes in the environment in which a system must operate. These environmental changes are normally beyond the control of the maintainer and consist primarily of changes to the: (1) rule, laws, and regulations that affect the system: (2) hardware configuration, e.g., new terminals, local printers, etc.: (3) data formats, file structures: and (4) system software, e.g., operating systems, compilers, utilities, etc.
- Perfective Maintenance – (Also known as enhancements and upgrades) All changes, insertions, deletions, modifications, extensions, and enhancements made to a system to meet the evolving and/or expanding needs of the user. It is generally performed as a result of new or changing requirements, or in an attempt to augment or fine-tune the existing software/hardware operations/performance. Activities designed to make the code easier to understand and to work with, such as restructuring or documentation updates

and optimization of code to make it run faster or use storage more efficiently are also included in the Perfective category.

Each of the following technical reviews generates a final documentation package delivered to the Government within 30 days of the event. The final package contains attendance lists, action items (AIs), disposition of the AIs, and any updates to material presented at the event in response to AIs.

Incremental Release Review**.** An Incremental Release Review (IRR) is performed during the requirements analysis phase of adaptive and perfective tasks when specified, and are attended by the Raytheon Team, the Government, DAAC staff, representatives of the science end-user community, and external interface representatives when applicable. Prior to the IRR, the task leader engages the SEIT and DAACs (or other stakeholders) in the analysis of the new requirement. The primary artifact of the analysis is a ticket for each new capability, containing the requirements, operations concept, derived requirements, transition considerations, high-level component requirements, operations and end user scenarios, and test verification criteria for functional, performance, and fault condition testing. The ARB appoints an engineer responsible for the generation of each ticket. Requirements analysis is a collaborative effort led by the responsible engineer who integrates the perspectives of the DAACs, NASA, and other stakeholders into the development of the ticket. Risks for each ticket are analyzed by the ARB. In addition, the task leader develops a task schedule that will be integrated into the program master schedule by the SEIT and presented at the IRR. Action items are recorded by the Task Lead. Action item status is reviewed by the PMT and retained for later use in the Release Status Review.

Consent to Ship Review. The purpose of the Consent to Ship Review (CSR), DID EMD-CSR-13, is to assess the readiness of the EMD Team and the DAAC to ship and accept the delivery. The CSR is performed when specified for major perfective changes that require a site readiness assessment prior to installation. For each CSR, a software and hardware physical configuration audit (PCA) is performed to ensure that there are no configuration discrepancies that might interfere with successful installation of the delivery. Action items are accepted from attendees of the review, consisting of DAAC operations and systems engineering staff and the Government. The Government grants approval to proceed only after the EMD Team has demonstrated satisfactory disposition of all action items.

Pre-Ship Review. The purpose of the Pre-Ship Review (PSR), DID EMD-PSR-14, is to review a final delivery package prior to its turnover to CM and subsequent delivery to the sites. The EMD Team performs a PSR for custom software patches and releases and COTS hardware and software upgrades. Installation instructions are reviewed for completeness by the DAAC staff. For most major adaptive and perfective releases, the PSR is the successor to the CSR. For other deliveries, which do not require the PCA associated with a CSR, the predecessor to the PSR is the installation, verification, and regression testing of the delivery at the EMD development facility. Action items are recorded from the PSR and retained for incorporation into the Lessons Learned Review (LLR).

Lessons Learned Review. The purpose of the LLR, DID EMD-LLR-15, is to provide the Raytheon Team, the Government, and the DAACs with a forum in which to improve the quality of future release support. An LLR will be performed as specified for perfective tasks. The

Raytheon Team will collect metrics following the deployment of a capability in order to measure the effectiveness of the PSR process. An example of such metrics is the number of requests for additional installation instruction information made by the DAACs following the PSR. The responsible engineer for the LLR will be a member of the PMT. LLR artifacts will be retained for use in planning future releases.

Release Status Review. The purpose of the Release Status Review (RSR), DID EMD-RSR-16, is to revisit the effectiveness of the performance of SDPS. The RSR will be performed annually unless a CSR has been performed in the last 12 months. Defects found after deployment, performance impacts, and impacts to DAAC productivity metrics will be measured and analyzed for application towards process improvements.

4.9 Project Process Improvement

Process improvement in SDPS custom software maintenance and development will be driven by the collection and analysis of metrics, and by the implementation of improvements and enhancements. Process improvements will be accomplished through the use of Raytheon's Six Sigma process improvement methodology.

A major source for ideas for process improvements is the Lessons Learned session, which is held after every significant software delivery (See Section 4.8). The results of the lessons learned activity typically leads to the initiation of Six Sigma efforts to resolve the major issues identified by the lessons learned activity. The Six Sigma Process improvement activity initiated from the lessons learned should be short enough in duration so that its results can be implemented prior to the next significant delivery

4.10 Software Quality Engineering

Software Quality Engineering (SQE) for the EMD Program will be performed by the Quality Assurance organization in accordance with the Software Quality Assurance Plan (SQAP). Quality Assurance will conduct process audits, product evaluations and monitor engineering, development, deployment, and maintenance activities. Examples of such activities include, but are not limited to program planning and tracking, design and development, hardware and software maintenance, configuration management, test, and release management. Quality Assurance will plan specific audits and evaluations for ongoing and maintenance tasks or identify QA schedule-related tasks based on the program schedule. More detailed information regarding the Quality Assurance organization, its structure, responsibilities, activities and processes can be found in the SQAP (104-EMD-001).

This page intentionally left blank.

5. Catalog of Services

EMD development services are summarized in Table 5-1. EMD offers a range of requirement, development, integration, and test approaches that can be combined in various ways to achieve cost-effective deployment of new capabilities. The services support development of new components or capabilities by the EMD contractor as well as other NASA stakeholders.

The services listed below are described in more detail in Section 6, Software Development Process, and Section 7, Corrective Action Process/Non-Conformance Reports.

Table 5-1. System Development Services enable cost-effective deployment of new capabilities (1 of 2)

Service Description	When to Use
Requirements Definition Service	
Provides SEIT services to coordinate hardware or software requirements definition across all stakeholders. Operation concept, requirements, and verification criteria are captured using the "Ticket" process. See Requirements Definition and Management below.	Medium to large capabilities developed by EMD contractor or community-developed capabilities where assistance in formal requirements capture is desired. Also used for interface definition or standards compliance activities where formal testing is to be done.
Requirements Management Service	
Incorporates new requirements into the verification database (VDB) and tracks verification status. See Section 6.2.1.	All EMD contractor developed capabilities where Tickets were generated. Optionally, for community-developed capabilities where requirements tracking is desired.
Operations Support Software (OSS) Development Service	
Perform development using a streamlined development process that relaxes requirements for formal documentation and formal verification. See Section 6.1.3.	Any non-core SDPS capability.
Incremental Development Service	
Perform development using an iterative process that includes a series of prototype or incremental deliveries of a capability to a DAAC or SIT. See Section 6.1.2.	Capabilities where requirements are not well known or where a capability could have significant operational impact and early feedback is warranted.
Formal Development Service	
Perform development using a waterfall process that includes formal peer reviews at preliminary design, detailed design, and code and unit test. Formal documentation is produced and formal verification is performed. See Section 6.1.1	Capabilities that involve modification of core SDPS functions. Other capabilities where rigorous design, development, documentation, and formal verification is required.
COTS Software Procurement, Integration, and Maintenance Service	
Perform procurement of new or additional COTS software licenses. Integrate COTS software products with existing hardware or software components. Perform COTS software upgrades when necessary and coordinate defect resolution with COTS vendors. See Section 8.	Any EMD contractor developed capabilities that are fully or partially implemented with COTS software products. Any community-developed capabilities where it is desirable to take advantage of the EMD contractor's buying power or where centralized COTS integration is beneficial.

Table 5-1. System Development Services enable cost-effective deployment of new capabilities (2 of 2)

Service Description	When to Use
Integration Service	
Perform integration of a new or enhanced capability with other SDPS components. See Section 6.8.	Any EMD contractor developed capabilities. Any community developed capabilities where it is desired to do a single integration rather than have each DAAC integrate.
Regression Testing Service	
Perform regression testing of a capability against SDPS to ensure that all SDPS functions operate correctly. If functional verification of capability was performed then regression test will also ensure the capability operates correctly. See Section 6.8.1.	Any EMD contractor developed capabilities. Any community developed capabilities where it is desired to do a single regression test rather than have each DAAC do regression testing.
Functional Verification Service	
Develop and execute test procedures to verify that a capability functions correctly. Optionally, conduct formal verification in conjunction with Government designated witnesses. See Section 6.8.3.	Any EMD contractor developed capabilities. Any community developed capabilities where it is desired to perform an external functional verification before deployment.
Performance Verification Service	
Develop and execute test procedures to verify that a capability meets performance and stability requirements under realistic SDPS workloads. See Section 6.8.2.	Any EMD contractor developed capabilities that are performance critical. Any community developed capabilities that are performance critical.
Configuration Management and Deployment Service	
Baseline and deploy a new capability. Test installation and transition procedures. See section 6.9.	Any EMD contractor developed capabilities. Any community developed capabilities where centralized configuration management and deployment is desired.
Training Service	
Provide installation, transition, and operations training for a new capability. This includes developing or procuring training materials and delivering training via training classes, train-the-trainer, or computer-based training. See Section 6.9.5.	Any EMD contractor or community-developed capability that requires instruction to be used effectively.
MR Tracking Service	
Coordinate collection, prioritization, and disposition of modification requests against a capability. See Section 7.1.	All EMD contractor developed capabilities. Any community developed capabilities where it is desired to a have a central coordination point for resolution of problems or enhancement requests.
Defect Resolution Service	
Provide corrective maintenance for a capability. See sections 7.2.	All EMD contractor developed capabilities. Any community developed capability where it is desired to use EMD resources for defect resolution.

6. Software Development Process

Figure 6-1 summarizes the software development activities of a typical capability and the artifacts produced from each stage. The activities are at the top of the bars and the artifacts below the bars. These peer reviews are required for software fixes and new capabilities exceeding SLOC thresholds as specified in the EMD Peer Review Process PI. Each phase contains a peer review (the milestone in the figure) of the outputs and a workoff period. The workoff period is where the defects from the peer review are resolved, resulting in improved artifacts.

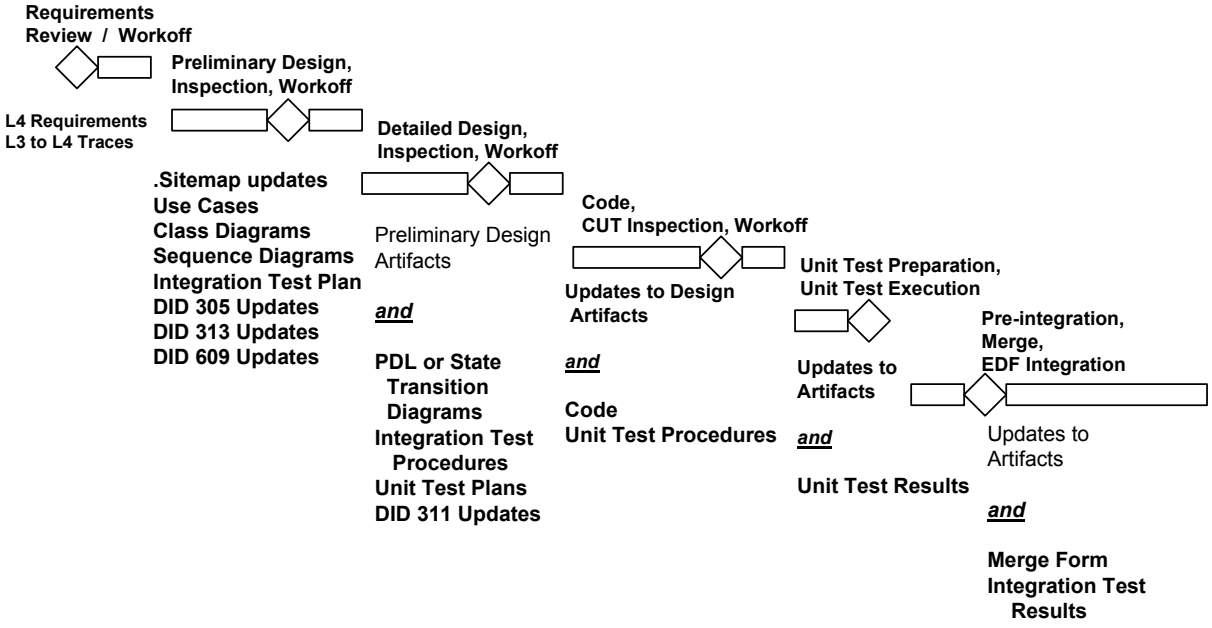

Figure 6-1. Development Tasks and Artifacts

6.1 Development Life-Cycle Approaches

Software development is driven either by the requirement to add a new capability (by virtue of a new EMD task order), or an enhancement or fix in existing operational code. In either case, the development organization has available to it several alternative software development approaches depending on the nature of the new capability, enhancement, or fix.

Three approaches are available Formal, Incremental and Operations Support Software (OSS). OSS is subject to a relaxed set of rules governing software development that are detailed in Section 6.1.3. Incremental and Formal development order their activities in a different manner, but both approaches are ultimately governed by the standard set of requirements for software development that are incorporated in Section 6 of this document.

These approaches support development of new components or capabilities by the EMD contractor as well as other NASA stakeholders. The determination of the approach depends upon the nature of the software requested.

6.1.1 Formal Development

This is a traditional waterfall development methodology that incorporates a peer review process after preliminary design, detailed design, and code and unit test as depicted in Figure 6-1. Formal documentation is produced and formal verification is performed. Each phase of the development (requirements, preliminary design, detailed design, code, unit test, and integration) consists of an activity, followed by a peer review and a work-off period for any issues discovered during the peer review. Each of the phases has a set of required artifacts that are specified in Development Program Instructions. All of the activities shown are scheduled and maintained in Primavera as part of the overall system schedule.

This approach should be typically used for capabilities that involve significant modification of core SDPS functions, or where rigorous design, development, documentation, and formal verification are required. Test procedures are developed based on the Ticket acceptance criteria. Formal verification is performed using test procedures that are developed from Ticket acceptance criteria and approved by NASA. Design, test, and operations documentation (DID 305, 313, 311, and 609) is produced in accordance with EMD-EDP-23, ECS SDPS Documentation Package.

6.1.2 Incremental Development

This is a development approach using an iterative process that includes a series of prototype or incremental deliveries of a capability to a DAAC or SIT. It can be utilized for capabilities and enhancements where requirements are not well known or where a capability could have significant operational impact and early feedback is warranted. The approach is intended to reduce the risk of substantial rework on software that may have significant operational impact at the DAACs, but this impact was difficult to assess during requirements and design peer reviews.

In this process, a subset of requirements is selected for initial design and implementation. The initial implementation is then deployed as a prototype to a pathfinder DAAC that has agreed to participate in early evaluation and feedback. The feedback is incorporated into the next design and implementation cycle, which improves the previous implementation and adds more requirements. Another prototype is deployed and more feedback is provided. This process continues until all requirements have been implemented. Each implementation cycle is typically short (2 to 3 months) with a 1-month evaluation period.

While the requirements and development standards are relaxed for the interim prototype development, the product ultimately delivered to the DAACs must be subjected to, and governed by, the software process requirements detailed in this document, including appropriate peer reviews, testing, and documentation.

6.1.3 OSS Development

This is a streamlined development methodology that relaxes the normal requirements for peer reviews, formal acceptance test, and the comprehensive documentation required by EMD-EDP-23. Typically, it is used for the development of scripts, Web-based applications, database applications, or stand-alone programs that extend SDPS functional capabilities or improve SDPS

operability by automating workarounds, routine maintenance tasks, system monitoring, or metrics collection. OSS components are standalone components that interface with SDPS by accessing SDPS databases, logs, external interfaces, or application program interfaces. Raytheon provides CM support for externally developed operational support software (OSS) components.

Three types of OSS components will be supported:

- Type 1 OSS components help automate EMD operations procedures (e.g., a SDSRV client driver that is used to support routine maintenance and recovery procedures). These components may be developed either by EMD or non-EMD organizations (e.g., non-EMD staff at the DAACs, NASA support contractors) and the operation of these components falls within the current scope of DAAC operations.
- Type 2 OSS components implement non-mission critical EMD Level 3 requirements (e.g., Whazzup). These components are developed by EMD using an operational prototyping process that relaxes requirements for documentation and formal verification that are not necessary or very cost-effective for this type of software. Use of this process to develop components that satisfy Level 3 requirements requires NASA approval. Operation of these components falls within the current scope of DAAC operations.
- Type 3 OSS components extend EMD requirements (e.g., EDGRS, QA MUT). EMD or non-EMD organizations may develop these components, however, EMD involvement in the development or operation of these components would require a CCR. Either NASA or Raytheon may identify a CCR as an OSS candidate and if both parties concur, the reduced costs associated with OSS development and maintenance will be reflected in the EMD ROM for the CCR.

The following requirements apply to the development of an OSS component:

- The component shall be developed with EMD supported languages and be able to be built with EMD baseline development tools (e.g., compilers, linkers, and script languages).
- The component shall be able to be executed with the baseline versions of EMD COTS (e.g., operating systems, command shells, DBMSs, web servers).
- The component shall be mode aware. That is, the component shall be easily configured to execute in any EMD mode and an instance of the component shall be able to be concurrently executed in each EMD mode.
- A component that is intended to meet EMD Level 3 requirements shall undergo the EMD Peer Review Process.

Component documentation shall be provided as README files, MS Word documents or HTML pages. The following minimum component documentation shall be provided:

- Component overview and operations concept;
- Release notes;
- Build instructions (if applicable);
- Installation instructions;

- Design documentation that describes: the purpose of the component and each of its subcomponents; how the component interfaces with EMD, including specifics on which database tables, log files, or APIs are used; and
- Component resource requirements, including platform where the component will execute; disk, memory, and CPU requirements; and types and frequency of accesses to EMD databases, log files or Application Programming Interfaces (APIs).

The component and subsequent updates shall be delivered to EMD in a specified directory structure in order to facilitate incorporation into the CM system. All necessary scripts to build, install, and deliver the component shall be provided by the developer.

A CCR is required to add a new OSS component to the EMD baseline. The Science and Development CCB will draw on EMD SEIT services, as necessary, to evaluate system impacts of executing OSS components within an EMD configuration.

Developers of OSS components that interface with EMD APIs will be required to access EMD API libraries through ClearCase views. The EMD Merge process will be required to incorporate these components into the EMD baseline.

OSS components are not subject to formal verification. It is the responsibility of the developing or maintaining organization to test new or updated OSS components prior to merging them into the baseline. It is the DAAC's responsibility to test OSS components after installation at the DAAC. EMD is responsible for maintaining OSS components developed by EMD staff. EMD is responsible for coordinating the maintenance of OSS components developed by non-EMD organizations. OSS component defects and enhancement requests will be managed using the existing EMD Trouble Ticket, NCR, Deployment IPT, and Priority Board processes.

Maintenance of EMD-developed OSS components will typically be performed by the EMD organization that originally developed the component (e.g., DAAC, Test, and Science Office). In order to reduce cost, a standing maintenance group will not be created for OSS components. Staff will be assigned to fix NCRs or make enhancements on an as needed basis, subject to current project priorities. Thus, turnaround time on NCR fixes and enhancements may vary. In the case where an OSS component was developed by a non-EMD organization, the Trouble Ticket or enhancement request will be forwarded by the SEIT for action.

6.2 Systems requirements development and management

The Functional and Performance Requirements Specifications (F&PRS) contains all of the L3 requirements that are to be maintained by EMD. Interface Requirements Documents and Interface Control Documents provide the interface requirements and specifications for EMD. The Verification Database (VDB) contains the L3 and Interface Requirements Document (IRD) requirements that have been decomposed and allocated to L4 requirements specific to EMD subsystem components. New capabilities can either add L3, IRD or L4 requirements, which require subsequent allocation and decomposition into components for design and implementation. The following requirement development process is described in the Requirements Management PI, and outlines how new capabilities are allocated to requirements.

Initially, SEIT allocates L3 requirements to architectural components including hardware configuration items and software configuration items, maps IRDs to L3s, and develops

operations concepts. Then, SEIT performs a detailed requirement analysis that includes working with Development to derive L4 requirements from the current L3 requirements. These L4 requirements are mapped to the L3s and IRDs requirements. SEIT then generates a set of verification tickets. These verification tickets are defined to group requirements (L3s, IRDs, and L4s) for logical testing and establish a complete set of Acceptance Criteria (AC) against which test cases should be evaluated to verify that these groupings of requirements are satisfied by the system. Thus, when all of the Acceptance Criteria in a ticket are verified, the ticket and its associated requirements are considered verified by association.

Peer reviews of the requirement tickets are conducted with stakeholders prior to capturing them in the Verification Database (VDB). Stakeholders include the representation from the applicable IPTs and CPTs, including representation from the DAACs. Following closure of action items taken at the peer review, the ticket is forwarded for final NASA review and approval. Subsequent to approval by NASA the ticket is baselined in the VDB. Any changes to these requirements require an approved CCR.

6.2.1 Software requirements management

A Ticket defines a system capability from multiple perspectives. It includes an operations concept, system requirements, and acceptance criteria. Requirements definition is led by the SEIT, working closely with all stakeholders. Tickets are defined by an SEIT architect, peer reviewed by all stakeholders, and approved by NASA. Once approved, the associated requirements and verification criteria are tracked in the VDB. Throughout the design, development, integration and test activities, the Ticket serves as the baseline definition for the capability and ensures all parties are working from the same specification. If requirement changes must be made during implementation, the SEIT architect will update the ticket, re-execute the review and approval process, and redistribute the updated specification to all parties.

6.3 Software Architectural/Preliminary Design

During the Preliminary Design phase, a high-level design is generated for each system capability. The preliminary design includes an almost identical set of design artifacts as detailed design, but the artifacts describe the design at a higher level.

The Rational Rose analysis and design tool is used to document the object-oriented artifacts of the design (object diagrams, sequence diagrams, use case diagrams, etc.). The Unified Modeling Language (UML) is the methodology used on EMD.

High-level descriptions of the artifacts generated during preliminary design are provided below. These examples are provided only to illustrate the types of artifacts produced during this phase. The most up-to-date list of preliminary design artifacts is documented in the PI for peer reviewing design. It is important to note that these artifacts apply to the new capabilities only. For example, use case diagrams do not exist for all the previous releases of EMD since UML was not the original methodology used on the program. Use case diagrams will not be produced for all the previous capabilities, only the new capabilities for a release.

- L4 requirements and their mapping to components of the design.
- HW/SW mapping showing what H/W components the S/W executes on.

- Object-oriented design artifacts such as: use case diagrams, class diagrams, sequence diagrams.
- Design deliverable documents: the Segment/Design Specification (305), Internal ICD (313), and Operators Tools Manual (609).
- Integration Test Plans
- SLOC and resource estimates. This is revised from the original estimates in order to review the feasibility of the implementation schedule.

Peer reviews are conducted to validate allocation of Level 4 requirements to the design components and to validate the overall high-level design itself. Potential candidates for software reuse are explored during this phase. The successful completion of the preliminary design peer review and the correction of all defects signify the completion of the preliminary design for that capability. The detailed design phase for that capability then begins.

6.4 Software Detailed Design

During the Detailed Design phase, a detailed "code-to" design is performed based on the preliminary design approved during the preliminary design peer review. In addition to the list of preliminary design artifacts, the following artifacts are generated. The most up-to-date list of detailed design artifacts is located in the PI for the design peer reviews.

- Program Design Language (PDL) to describe the complex algorithms of methods. Guidelines for which methods require PDL are included in the PIs.
- State Transition diagrams can be developed as an alternative to the PDL.
- Fully populated object-oriented design artifacts. At preliminary design the object-oriented artifacts are at a high-level. At this stage, they are fully defined. For example, the class diagrams will include all classes with all attributes and their data types and all methods with full signatures.
- Updates to the Database Design and Database Schema Specifications (311)
- Integration Procedures are produced providing detailed steps on how to integrate the final software capability. These Integration Procedures are incorporated in the Acceptance Test Procedures by the Test and Integration Team.

Peer reviews are conducted to validate the detailed design of the capability. After completion of all the defects identified during the peer review, the detailed design phase is complete and implementation can begin.

6.5 Software Documentation

During the preliminary and detailed design phase, the software deliverables referenced in Table 6-1 are generated. These deliverables will consist of redlines or change pages of the existing documents. After implementation and integration, these redlines are incorporated into a full document set of “as-built” documentation, and also comprise a portion of the EMD-EDP23 CDRL.

Table 6-1. Software Design Documentation

Document #	Document Name	Description
305/DV2	Segment/Design Specification	Provides details on the context and design of CSCIs, at the Unix process level. In addition, information about libraries and classes are provided for pointers into the code.
313/DV2	Internal ICDs	Provides details of interfaces between CSCIs including protocol information. Scenarios are used to illustrate interfaces. Tables provide high-level information about the interfaces such as whether they are remote procedure call interfaces or low-level socket calls.
609/DV2	Operations Tools Manuals	Provides details of operator tools (the graphical user interfaces). Each operator tool is explained without regard to the procedures being operated. Other documents discuss the procedures used to perform functions of an operator.
311/DV2	Database Design Specification	Provides information about the database tables, columns, relationships, indexes, etc. Includes everything associated with the physical implementation of databases in the system.

6.6 Software Code and Unit Test

During software coding, classes are coded and a clean compilation produced. Software Engineers follow coding standards and naming conventions PIs during this process. A different coding standard is provided for each language used on EMD. Developers use the Unit Test Plan and Execution PI to develop a set of step-by step unit test procedure to verify that the requirements are satisfied. Code & Unit Test Peer Reviews and artifacts are described in the Code and Unit Test Peer Reviews WI.

After the code is peer reviewed and the defects from the peer review resolved, the code is merged to the baseline. The merge process is documented in the Merge Process PI, but is summarized here at a high-level.

- A merge request is submitted to the Software Turnover Tracking System (STTS) upon completion of the unit test.
- The merge request is discussed at a meeting with all the subsystems represented
- If the merge request is complete and the integration lab is ready to integrate the functionality, the merge is approved.

- Upon approval, the software developer responsible for the code, uses the configuration management tool ClearCase to "merge" the code to the appropriate software release baseline.

The "merge" of the code to the appropriate release baseline signifies the end of the implementation phase and the beginning of the integration phase. The software is built by SCM and staged to an area within ClearCase where the integration lab can receive it.

6.7 Software Integration and Test

Custom Code Maintenance is responsible for integrating the software into a working software system, through the execution of integration procedures. The Software Integration and Test Lab (SWIT) oversees the integration. Problems with the software are resolved through an iterative approach of writing NCRs, fixing them, merging them to the baseline, and updating verifying the NCR fixes. The NCRs that are written are considered "informal" NCRs since they are found internally. The software development project instruction for NCRs defines the process for these NCRs from the time of submittal through resolution including the definition of the severity of the NCRs.

When all the severity 1 and 2 NCRs have been resolved for a particular integration procedure, a formal run is executed. SWIT oversees the procedure execution. Quality Assurance, SEIT and Test (SIT) are invited to attend. The results are documented on a test execution form. Only non-critical NCRs can exist in order for the integration procedure to be considered a successful execution. These non-critical NCRs are recorded in the NCR database and on the test execution form. The test execution form is saved in a test execution folder. When all of the required integration procedures for a release have been successfully executed, a System Integration Readiness Review (SIRR) is conducted. The SIRR will ensure that the software tested meets the corresponding Level 4 requirements, and the integration test documentation is complete. SWIT provides direction to SCM to generate a code baseline and tar file(s) of executables prior to the meeting. System Integration and Test (SIT) determines the readiness of the product based on the items supplied by SWIT.

6.8 Testing Approaches

The EMD team uses a multi-layered approach to testing; the type or level of testing is scaled to match the type of software delivery. The EMD deliveries include: Engineering Software (ES), Test Executable (TE), Patch, and Release. Delivery mechanisms for each of these software deliveries is discussed in section 6.9 Deployment Options.

6.8.1 Regression/Fault Recovery Test

The purpose of Regression Testing and Fault Recovery Testing is to exercise the major functions of EMD to provide confidence that the addition or modification of custom or COTS software does not adversely affect the behavior of unmodified code. Fault Recovery testing ensures that EMD software responds properly to server outages and other anomalies under various conditions. The Regression Test Plan provides an overview of the methodology used for the selection, development, and execution of Regression Test activities.

Regression Test activities are based on normal production scenarios that will exercise EMD functionality. These activities tailored to each facility and contain the following:

- Test Checklist - The purpose of Test Checklists is to provide a list of functional system threads for each subsystem
- Representative sample of tests that will exercise software functions
- Additional tests that focus on software functions likely to be affected by a new release/update
- Tests focusing on software components that have changed

The regression activities are based on the functional threads found in the checklist. These threads will compose a scenario beginning with ingest, archive, and production, and ending with search, order, and distribution. This scenario is designed to test the basic functionality of the system after a release or patch is installed. By running this test each time, expected results form a baseline for future regression testing of the system. In addition to the Insertion-Production-Retrieval scenario, several other threads will be developed based on related functions not tested in the core scenario. These threads will be tested only if the new functionality may affect it.

Finally, new functions that are delivered with each new drop or patch will be analyzed, and a determination will be made as to which components could be affected by the new software. Existing regression test cases will be updated to include the new functionality.

Regression testing will be performed after each new software release. Regression testing will also be performed at the DAACs after installation and checkout of the S/W after CSR. These regression tests will be tailored to include test cases that exercise specific capabilities of interest to the DAAC, in addition to the general capabilities of the software.

6.8.2 Performance Verification Testing

The Performance Verification Center (PVC) was established to test the system under a load equivalent to the load that will be present during operations. Special test procedures are defined to test the performance and stability of the system under operations loads. Parallel to the system being tested against the functional requirements in the VATC, the system will be tested against performance criteria in the PVC. NCRs are generated for performance issues just as they are written for functionality issues. Patches are applied and the performance regression tested until the release performs satisfactorily in order to deploy. This must be accomplished prior to the CSR of a release.

Performance verification on system-wide releases typically consist of executing at least two 24 hour sustained operation tests using workloads that approximate the required loads for the release deployment timeframe. They are normally preceded by 24 hour dry runs intended to allow test engineers to resolve problems with scenario execution and configuration. These tests will be self-contained and not use external interfaces. The success criteria will be the demonstrated ability of the system to execute each workload within a 24-hour period.

For a typical release, GSFC and EDC DAAC workloads have been selected since they have the largest throughput rates. The workload specification has been derived from the SOW and F&PRS requirements for mission support, capacity phasing, and catch-up rates. The workloads use granule sizes, granule counts and PGE execution frequencies as defined in the EMD

technical baseline. Workload specifications are periodically updated to reflect more closely typical DAAC loads. At its discretion, EMD will use synthetic data/PGEs, real data/PGEs, or a combination to implement the workload.

The PVC often cannot provide sufficient capacity to fully test performance in all areas. The likely areas of reduced capacity are:

- Fewer science processor CPU's and processing disks than the largest DAAC;
- Fewer silos and archive tape drives than the largest DAAC; and
- Fewer physical media distribution devices than the largest DAAC.

In this case, the workload specification will be adjusted to be consistent with the PVC hardware and network capacity. For example, PGE execution times may be shortened to permit execution of full processing chains on a smaller number of CPU's. This approach will permit verification of the system's ability to plan and schedule the required number of PGEs per day on a smaller science processor configuration than would be required if baseline PGE execution times were used. Any changes required to the workload specification will be identified in an update to this document.

In order to reduce hardware, test development and execution costs, performance verification of secondary and/or low rate functions will not be performed. These functions include: failure recovery, failover, user registration; user profile update; user profile replication; user login; DAR submit; DAR status; directory search; GDS gateway requests; expedited data processing and distribution; ancillary data ingest from minor sources; and operator functions not related to core ingest, archive, production, and distribution.

6.8.3 Acceptance Test

At the completion of the installation and checkout of a release, a Test Readiness Review (TRR) is held. The purpose of the TRR is to assess the readiness for the start of acceptance tests. The problems found in installation and checkout are assessed as well as the readiness of the acceptance test procedures. The pre-installation serves as a pathfinder for the installation of the formal delivery occurring after CSR.

Acceptance testing consists of executing operational scenarios on the system. Acceptance test procedures are developed during design and implementation, debugged during integration, and approved by SED and the customer. The System Verification group, to the maximum extent possible, establishes representative site configurations within the VATC or PVC to verify the site-unique testing to be performed in the field.

After TRR, the System Integration and Test group begins to execute the acceptance test procedures. Each procedure is dry run first. After successful dry runs of the procedure, a formal run is scheduled with IV&V and the customer. Quality Assurance also witnesses formal runs of the tests on a sampling basis. Upon completion of the full set of acceptance tests, a Consent to Ship (CSR) review is held and the software is shipped and installed in the field (the DAACs).

Problems found during acceptance testing are documented in NCRs. The Custom Code Maintenance IPT resolves the NCRs and supplies patches to the release as requested by the SIT group. Refer to section 7 of this document for more discussion on how NCRs are resolved.

6.9 Deployment Options

The Raytheon Team understands from its ECS experience that delivery plans need to be flexible and clearly communicated. The Raytheon Deployment IPT will communicate with all stakeholders about the delivery plans for all system changes. This communication is provided via teleconferences, e-mail, and the Monthly Patch Plan. The Deployment IPT (DPT) briefs status updates to deployment plans during the MRB; longer patch planning sessions with the DAACs are held periodically via teleconference. Delivery planning and distribution notices are sent to a mailing list of deployment stakeholders maintained by the DPT. Finally, all deployment plans are documented in the Monthly Patch Plan (MPP), which provides a 3-month look-ahead schedule for all deliveries to the field; the MPP is updated weekly (despite its name) in order to keep it current with changes in the Priority List and DAAC needs. The EMD program will use a Priority List to determine work assignments. Section 7 details how the Priority List is derived.

The Raytheon Team provides a flexible approach toward SDPS custom software deliveries that provide small, timely fixes for urgent problems and larger subsystem roll-up patches for the delivery of less time critical fixes. Full system deliveries are driven by integration activities that require the establishment of a new maintenance baseline.

6.9.1 Engineering Software

Engineering software is software delivered to a DAAC in response to an emergency. It is only provided at the specific request of the DAAC. The goal of engineering software is to provide mitigation of a critical problem in DAAC operations within hours of the onset of the problem. Engineering software may be delivered at any time of the day or day of the week. Engineering software is built in a developer's view from source code that is not yet merged to the baseline and typically only unit tested by the developer. Our goal within EMD sustaining engineering will be to merge corresponding fixes (tested and approved) to the appropriate baseline within 48 hours of sending the engineering software. The Raytheon Team will continue to use a custom ClearCase tool developed under the ECS contract that controls and documents the delivery of engineering software to the field. This tool, accessible only by senior developers (subsystem leads) who are authorized to send engineering software, captures and documents what engineering software has been sent. Engineering Software Delivery instructions are defined in the Engineering Software Delivery PI.

6.9.2 Test Executables

TEs are software deliveries designed to fix a specific problem with the smallest possible delivery footprint. The footprint of a change is the number of delivered components required to implement the change. The goal of a TE is to provide a fix for an urgent problem as soon as it is available (possibly before final, definitive testing is completed), in a form that can be promoted into operations by the DAAC as quickly as possible. TEs are also only provided at the specific request of the DAAC. TEs are normally delivered during the standard work week, but can be built and delivered during off hours for especially urgent problems. TEs are delivered from the controlled baseline (after a merge and build), either from the maintenance baseline or from an Emergency Bug Fix (EBF) branch. TEs are delivered under a configuration change request (CCR) routed through the EMD Science and Development CCB, in tar files prepared by software CM. Software CM uses SDPS custom tools to send the TE tar files, document the receipt of the

delivery by each site, and distribute all of the technical data (NCR lists and installation instructions) to all interested parties. TEs are installed and tested by the test group in the test environment (VATC and/or PVC), but this testing may be done in parallel with the delivery to the DAAC depending on the urgency of the request. Under EMD, the delivery of TEs will be planned and managed on a daily basis by the lead of the DPT. The Custom Software Delivery PI governs the delivery of Test Executables as well as Patches.

6.9.3 Patches

Patches are larger software deliveries, usually covering an entire SDPS subsystem. The goal of a patch is to deliver all of the fixes that have been applied to the maintenance baseline for an entire component or subsystem. Sometimes the fix to a problem will require the simultaneous delivery and installation of components from multiple SDPS subsystems; when this happens, multiple subsystem patches are generated, tested, and delivered as a group. Patch deliveries are planned and scheduled by the Deployment IPT, taking into account the program priority list, the list of program mission milestones, development progress in merging fixes for specific problems, and inter-relationships between fixes among subsystems. Under EMD, we will provide a subsystem patch for each major SDPS subsystem every 6 to 8 weeks. Patch delivery follows a rigorous process. The Software Integration and Test Team sends a request to software CM to build tar files for the patch, identifies the NCRs fixed in the patch, and generates draft installation instructions. The test group tests the installation of the patch in the PVC or the VATC, providing redlines to the installation instructions as required, and verifies the NCRs fixed in the patch. Additional functional regression testing is performed as required. When testing is completed, the patch is presented at a pre-ship review by the DPT. The installation instructions and NCRs are discussed with the DAACs, including any changes in operational procedures or troubleshooting methods required by the fixes. A CCR is executed to deliver the software, and software CM uses SDPS custom tools to send the patch tar files, document the receipt of the delivery by each site, and distribute the technical data package for the delivery.

New and modified Earth Science Data Types (ESDTs) will be delivered as special, low-overhead patches. These patches will be specially packaged to be independent from all other subsystem changes so that they can be installed immediately. Ingest database updates and valid values updates, if required by the ESDTs, will be delivered with them and will be installable independent of all other subsystem patches.

6.9.4 Drop/Release

The delivery of the full SDPS system is usually referred to as a drop or a release, and is accompanied by the transition to a new maintenance baseline. Full system deliveries will be performed only as necessary to deliver capabilities and upgrades that require changes in a majority of the subsystems. Extensive regression and performance testing precede full system deliveries.

6.9.5 Transition/Training

Occasionally, deliveries will require a set of transition activities to be performed (e.g., building of a database index or updating values in a database table) either before or after installation of the software. Scripts are provided to perform these activities and are included in the installation

package along with appropriate instructions. Sometimes full system deliveries require a complex set of transition activities. When this occurs, a detailed transition plan is developed and the DAACs are invited to the Landover facility for transition training and hands-on transition exercises in the VATC or PVC. A pathfinder DAAC is selected to receive an early delivery of the software. With proactive support from EMD engineers, the pathfinder DAAC will test installation and transition. Feedback from this exercise is incorporated into installation and transition instructions prior to deployment to all DAACs.

For new capabilities not requiring the scope of support described above, the primary development engineer is invited to one of the bi-weekly deployment IPTs to discuss the new capability.

6.10 Process Variances

In order to diverge from the software development process, a process variance request must be approved. A project instruction describes the process for this and the signature approval authorities. This provides a mechanism to document and heighten awareness within the project when a process does not make sense for a particular instance. Process variances also provide a means to determine process improvements. If process variances are frequently being approved for the same part of the process, then it may be time to change the process.

For example, an NCR fix that is stopping all archival of a particular kind of data at one DAAC has a 225 SLOC change. The peer review process requires a three-day inspection notification to peer review the code change. A process variance request could be approved to reduce the three day inspection notice to one day. An alternative process variance request could request to use the walkthrough method instead of the inspection method, which only requires a one day notice.

This page intentionally left blank.

7. Corrective Action Process/Non-Conformance Reports (NCR)

The EMD Program uses NCRs to document system problems and their resolution. Each NCR contains detailed information on the problem, including but not limited to, the problem description, problem submitter, where the problem was found, the severity of the problem, and the priority with which the problem should be fixed. In addition, the NCR contains information provided by the developer, including but not limited to, the status of the fix (e.g., under analysis, merged, in test), the estimated size of the fix, the estimated merge date, test information, and so on.

7.1 Prioritization of NCRs

To ensure that all DAACs' most important operational issues are addressed, EMD uses a Priority List to determine which operational issues should be addressed. The Priority Review Board Process PI details the process that is employed to generate the weekly priority list. At a high level the process is as follows:

1. The Deployment IPT receives a top priority list from each DAAC and posts them at http://smc.gsfc.nasa.gov/public_html/ipt/ipt_top10.html.
2. The Operations Deployment IPT Lead or designee receives updates/changes and new inputs for NCRs found in the operational baseline during internal testing (VATC, PVC). Inputs may also be submitted by the Custom Code leads, Science Office (new or updated ESDTs) and SEIT. Inputs can be received from various meetings (i.e., the Daily Merge meeting) and via email.

   3. Using various factors as detailed below, the Operations Deployment IPT lead or designee generates the Priority list and posts it on the web site above.

   - Include the top five NCRs from each DAAC.
   - Include the top five NCRs found during the development and testing phases in the EDF
   - Include other NCRs as directed by NASA and Systems Engineering.

7.2 NCR Process

The NCR process is detailed in the Development Planning and Tracking of Operational NCRs PI. At a very high level the process is as follows:

1. DAACs submit trouble tickets to the help desk.
2. After trouble tickets are converted into NCRs, they may be prioritized.
3. If an NCR is prioritized, an initial assessment is made of the NCR.
4. A resource is assigned to work on the NCR by the subsystem lead.

5. The developer, with the aid of the subsystem lead, analyzes and estimates SLOC, effort and schedule.
6. If the NCR is a severity 5 (a request for a new capability) and exceeds 80 hours of effort it is forwarded to SEIT for review; otherwise, the developer proceeds to fix the NCR.
7. Developer and technical lead perform a technical review of proposed merge.
8. Developer submits NCR for merge.
9. The merge form is reviewed at the Daily Merge Meeting.
10. If approved, the developer merges NCR.
11. The developer validates the fix in the EDF.
12. DDTS is updated with Actual SLOC, Effort, and Schedule.
13. Using the Patch Schedule, a tar file is generated for the SIT group.
14. Upon successful testing on the patch, the Deployment IPT delivers the patch to DAACs.

8. COTS Maintenance/Insertion

The EMD COTS software maintenance technical approach addresses the full COTS product life cycle. It includes DAAC participation in each phase of the life cycle, and it adapts based on customer schedule and requirements, risk mitigation, lessons learned feedback, continuous measurable improvement, and metric analysis. COTS Upgrade Team (CUT) meetings are held weekly to review COTS software upgrade needs, plans, schedules, and status. Participants include representatives from engineering and support teams (including Quality, Test, Security and CM) as well as the customer.

COTS software maintenance encompasses:

- Identification of requirements
- Selection of COTS products to meet requirements
- Upgrade and maintenance of COTS products, driven by factors such as:
  - End of Life/End of Support of software versions
  - Changes to other COTS software versions, custom code implementations and hardware platforms
  - Product replacement or obsolescence, sometimes caused by vendor acquisitions and mergers
  - Security issues related to COTS products or COTS product implementation
  - Problem resolution
- Risk mitigation relating to overall COTS product deliveries for the EMD Project requirements, schedules and costs
- Retirement or replacement of COTS products because of changes in requirements and/or technology evolution (technology insertion).

Detailed information on COTS upgrade plans are made available to the DAACs in the following documentation:

- EMD COTS Deployment Plan (DID335) - provides a long term outlook for COTS upgrades planned. The plan details the COTS life cycle characteristics identified above.
- Deployment Patch Plan - provides a monthly update on COTS product schedules and deliveries.
- COTS Compatibility Matrix - provides a weekly update on the metrics of COTS products deliveries, including Primavera schedule summaries. Details of issues that are identified in the upgrade process are also identified on a weekly basis.

The sections that follow describe the key processes, tools, configuration control methods, and metrics used for performing routine and emergency COTS software maintenance.

8.1 Key COTS Software Upgrade Processes

COTS software upgrades are carefully selected, timed, and implemented to balance needs, risks, and costs. Up-front analyses precede controlled integration and regression testing in a variety of test environments, and deliveries to sites include installation and transition instructions used and proven at Landover. The upgrade process consists of six phases.

8.1.1 COTS Software Upgrade Analysis Phase

The COTS software upgrade life cycle begins with COTS software upgrade analysis. The COTS Upgrade Team continually evaluates COTS product vendor changes and assesses product version improvements, stability, compatibility, and impacts. This phase of the life cycle addresses:

- Long Range Planning: A COTS Compatibility Matrix is maintained to provide relevant data for managing the full COTS product life cycle and for mitigating COTS product-related risks to EMD custom code maintenance. For example, the information in this database was used to develop the staggered upgrade plan for the major Solaris 8 release, mitigating risks that might have occurred had all COTS products been upgraded at the same time.
- Planning: As part of the COTS upgrade life cycle, the COTS Upgrade Team prepares a planning document with distribution to all stakeholders, including Custom Code and Operations Deployment teams. This document includes all relevant Compatibility Matrix information so that issues can be identified and worked at the earliest stage possible. Action items are assigned and resolved prior to initiating work on the upgrade.
- Bundling: Bundling of COTS products—the delivery of upgrades to several COTS software products in a single, integrated package—provides efficiencies under certain circumstances. This is most effective when products to be bundled are "related" in some way and when benefit can be derived by installing and testing them together.
- Technology Refreshment/Insertion: When a major future change in a COTS product is identified, senior managers are presented evaluations on which to base decisions and recommendations. These studies help reduce the risks and impacts that can result from COTS software obsolescence. They also offer opportunities to reduce overall program maintenance costs. Evaluations consider:
    - Cost effectiveness of upgrading the COTS product compared to replacing it with a different COTS product or with custom code.
    - Impact of hardware upgrades on software upgrades and vice versa.
    - Impacts of product obsolescence
    - Continuity of vendor support for an EMD hardware platform
    - If additional licenses or other procurements may be necessary.

The COTS Upgrade Team also coordinates upgrade issues and actions, and acquires the COTS software media, documentation, required licenses and licensing keys, and supporting software for upgrades. All COTS software media is controlled via a COTS software library process (see

Section 4.6.3). COTS software-related custom code modifications are incrementally delivered as part of custom code software releases that are backward and forward compatible.

8.1.2 Readiness and Planning Phase

Readiness and Planning activities ensure the upgrade can be accomplished successfully within the required time and with available resources. DAAC participation during planning, through regular telecons and briefings, ensures consideration of operational impacts.

A key activity in this phase is determining the COTS upgrade strategy, which can be standard or fast-track. The fast-track process reduces analysis, planning, engineering and/or testing without reducing quality. The fast-track process tailors standard procedures for COTS products that do not impact the operational system (e.g., Netscape Communicator or Purify), for products unique to certain environments, (e.g., AMASS is only available in the PVC and VATC and not the COTS Evaluation and Software Integration Labs) or when the upgrade is performed at a DAAC prior to general deployment. In these cases, analysis, engineering and testing in selected environments is redundant and, therefore, unnecessary to successfully deploy a quality COTS software product. The standard upgrade strategy is based on the COTS resources needed (e.g., integration and test environments), testing needed, complexity of the product, and footprint of the product in the SDPS system.

Another key activity is determining the transition approach, including data conversion and data migration requirements, which must be factored into the upgrade strategy. The amount of COTS testing needed is dependent on the product For example, OS upgrades require extensive testing, while a Web browser upgrade needs less testing.

Other key elements of this phase include: needs analysis; identification of enhancement and new capability aspects of the upgrade; commitment of personnel and system resources; task decomposition, ownership, and completion dates; clearly defined integration, regression, performance, and fault recovery testing objectives for all environments; and mitigated risks and dependencies associated with the upgrade. A transition approach is developed in participation with the DAACs to maximize site upgrade effectiveness and minimize site impacts.

This phase results in an Evaluation Plan, detailed schedule, and internal review of the plan for completing the upgrade.

8.1.3 COTS Software Engineering Phase

COTS software engineering forms the core of the COTS software upgrade mechanics: design, modification, verification, and documentation of COTS software configurations and custom code modifications.

Installation and testing in the COTS Evaluation Lab ensures successful implementation of major COTS software features and identifies early custom code impacts and solutions, thereby maximizing downstream implementation, testing, and operations effectiveness. COTS Evaluation Lab testing also demonstrates that major features of the upgraded COTS software product work correctly and do not adversely affect SDPS custom code.

After testing in the COTS Evaluation Lab, the responsible engineer prepares a Pre-Ship Review (PSR) document (EMD-PSR-14). This document contains installation instructions, configuration

and environment parameters, transition and data migration procedures, test procedures and results, and any errors (NCRs) that were detected and their solutions.

Next, Software Integration and Test Lab installation and testing ensures successful COTS software integration with the SDPS custom code. The COTS Software Maintenance and Infrastructure Support teams collaborate on the best way to install and configure the COTS software product, paving the way for smooth COTS software integration at the PVC, VATC, and DAACs. The Custom Code Maintenance team tests the SDPS custom code with the COTS software upgrade, verifying that the upgrade's installation and configuration procedures support new or existing SDPS custom code functionality and that the COTS software features work as needed.

Upon successful completion of integration testing, the responsible engineer updates the PSR document, capturing changes to test procedures and results, install instructions, and NCRs and resolutions, as well as other appropriate documents and training materials.

8.1.4 COTS Software Verification Phase

Formal verification of COTS software is performed in the VATC and PVC to ensure the quality of the COTS installation and configuration procedures for DAAC-like environments, as well as full functionality in an operational environment. The Test team conducts tailored, system-level integration, acceptance, performance, regression, and fault-recovery testing to ensure DAAC operational functionality, stability and performance. Transition testing and necessary training with the site ensures operational installation, configuration quality, and minimal DAAC operational impact. Once again, PSR documentation is updated with changes to test procedures, test results, install instructions, and NCRs and resolutions.

8.1.5 COTS Software Review and PSR Phase

Following successful verification testing, the Operations Deployment team coordinates a series of reviews. An internal review of all COTS software upgrade materials verifies quality, accuracy, completeness, and compliance with approved processes and customer needs. Next, a review with the customer – the DAAC Walkthrough – provides a forum to discuss aspects of the product release in detail and to answer any remaining site configuration, upgrade and transition questions. The PSR is the final review, at which approval of the COTS software upgrade, PSR documentation, CCR, and baseline changes are received.

8.1.6 COTS Software Deployment Phase

Once the COTS software PSR completes, the Configuration Management team updates the baseline and distributes the COTS software media and documentation to the sites. The Operations Deployment and COTS HW and SW Maintenance teams provide customer support during installation at the sites, ensure the upgrade gets installed in remaining Landover environments that need them, and collect customer COTS software upgrade lessons learned and feedback to ensure continued process improvement and customer satisfaction.

8.2 COTS Test Executables (TEs)

DAACs occasionally require delivery of an emergency COTS product upgrade before full evaluation and testing can be completed. In response to an approved, high severity NCR or

trouble ticket, the COTS HW and SW Maintenance team obtains the product and coordinates whatever checkouts and tests the urgency of the fix allows. They then provide the product and installation instructions to CM for forwarding to the sites under an approved CCR. COTS TEs are used at risk.

8.3 COTS Problem Resolution

Staff at Landover and support contracts with COTS product providers combines to ensure COTS product-related problems are resolved. EMD sites or users experiencing problems with COTS products receive assistance by reporting the problem via EMD trouble ticketing system or via phone or e-mail to the Landover Help Desk.

Any problem that cannot be resolved by the Help Desk is prioritized in coordination with the sites and forwarded to Landover's COTS Maintenance team. If a product defect is the likely cause, an NCR is generated that can be traced to the trouble ticket. Team members escalate problems they cannot resolve themselves to the product vendors. Solutions may be a temporary workaround, patch, or product upgrade.

Progress on COTS product trouble tickets and NCRs is tracked continually. Status and priority are reviewed with the sites at least weekly using the same process as for custom code.

8.4 COTS Software Tools and Configuration Control Methods

Table 3-1. lists the tools used to support EMD maintenance and development. Key tools for the COTS software processes are the COTS Compatibility Matrix, Primavera, and ClearCase. Configuration control methods are described in Section 4. Key CM activities related to COTS software maintenance and development are the management of COTS software media and licenses, delivery of software products to the customer, and management of the COTS software configuration baseline for each facility.

8.5 COTS Software Metrics

Section 4.4 identifies the software metrics in use for EMD. Key metrics for COTS software activities are: installation NCRs for COTS; patch installation (i.e., the time to install standard and fast-track COTS software patches into operations); schedule performance; and cost performance.

This page intentionally left blank.

Abbreviations and Acronyms

AI	Action Item
API	Application Programming Interface
ARB	Architecture Review Board
BOE	Basis Of Estimate
CCB	Configuration Control Board
CCR	Configuration Change Request
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CM	Configuration Management
CMM	Capability Maturity Model
CMP	Configuration Management Plan
COTS	Commercial Off-the-shelf
CPT	Cross Product Team
CSCI	Computer Software Configuration Item
CSR	Consent to Ship Review
CUT	COTS Upgrade Team
DAAC	Distributed Active Archive Center
DCN	Document Change Notice
DDTS	Data Defect Tracking System
DID	Data Item Description
DLOC	Delivered Lines Of Code
DMO	Data Management Office
DPT	Deployment IPT
EBF	Emergency Bug Fix
ECS	Earth Observing System Data and Information System (EOSDIS) Core System
EDF	Engineering Development Facility
EMD	EOSDIS Core System (ECS) Maintenance and Development
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ES	Engineering Software
ESDT	Earth Science Data Type
F&PRS	Functional and Performance Requirements Specifications

FCA	Functional Configuration Audit
GSFC	Goddard Space Flight Center
HMDP	Hardware Maintenance and Development Plan
IPDS	Integrated Product Development System
IPT	Integrated Product Team
IRD	Interface Requirements Document
IRR	Incremental Release Review
IV&V	Independent Verification and Validation
LLR	Lessons Learned Review
MCMR	Monthly Contractor Manpower Report
MPP	Monthly Patch Plan
MPR	Monthly Progress Report
MR	Modification Request
MRB	Modification Review Board
NASA	National Aeronautics and Space Administration
NCR	Non-conformance Report
OSS	Operations Support Software
PAR	Performance Assurance Requirements
PCA	Physical Configuration Audit
PDL	Program Design Language
PEB	Performance Evaluation Board
PI	Project Instruction
PMR	Program Management Review
PMT	Program Management Team
QA	Quality Assurance
QAE	Quality Assurance Engineer
RC	Risk Coordinator
RI	Responsible Individual
RMP	Risk Management Plan
RSR	Release Status Review
SCM	Supply Chain Management
SCA	Subcontractor Administrator
SDF	Software Development Folder
SDL	Software Development Library
SDP	Software Development Plan
SDPS	Science Data Processing Segment

SEIT	System Engineering And Integration Team
SEL	Software Estimation Lead
SEP	System Enhancement Proposal
SEPG	Software Engineering Process Group
SIRR	System Integration Readiness Review
SIT	System Integration and Test
SLOC	Source Lines of Code
SMC	System Monitoring Center
SMDP	Software Maintenance Development Plan
SOIs	Software Operating Instructions
SPR	Subcontractor Progress Review
SQAP	Software Quality Assurance Plan
SRA	Site Readiness Assessment
STTS	Software Turnover Tracking System
SWIT	Software Integration and Test
TE	Test Executable
TPR	Task Plan Request
TRR	Test Readiness Review
UML	Unified Modeling Language
VATC	Verification And Test Center
VDB	Verification Database
VDD	Version Description Document
WI	Work Instruction

This page intentionally left blank.